Soft Comput (2009) 13:307–318
DOI 10.1007/s00500-008-0323-y

FOCUS

KEEL: a software tool to assess evolutionary algorithms for data mining problems

J. Alcalá-Fdez · L. Sánchez · S. García · M. J. del Jesus · S. Ventura · J. M. Garrell · J. Otero · C. Romero · J. Bacardit · V. M. Rivas · J. C. Fernández · F. Herrera

Published online: 22 May 2008

Abstract This paper introduces a software tool named *KEEL* which is a software tool to assess evolutionary algorithms for Data Mining problems of various kinds including as regression, classification, unsupervised learning, etc. It includes evolutionary learning algorithms based on different approaches: Pittsburgh, Michigan and IRL, as well as the integration of evolutionary learning techniques with different pre-processing techniques, allowing it to perform a complete analysis of any learning model in comparison to existing software tools. Moreover, KEEL has been designed with a double goal: research and educational.

Supported by the Spanish Ministry of Science and Technology under Projects TIN-2005-08386-C05-(01, 02, 03, 04 and 05). The work of Dr. Bacardit is also supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant GR/T07534/01.

J. Alcalá-Fdez (✉) · S. García · F. Herrera
Department of Computer Science and Artificial Intelligence,
University of Granada, 18071 Granada, Spain
e-mail: jalcala@decsai.ugr.es

S. García
e-mail: salvagl@decsai.ugr.es

F. Herrera
e-mail: herrera@decsai.ugr.es

L. Sánchez · J. Otero
Department of Computer Science,
University of Oviedo, 33204 Gijón, Spain
e-mail: luciano@uniovi.es

J. Otero
e-mail: jotero@uniovi.es

M. J. del Jesus · V. M. Rivas
Department of Computer Science,
University of Jaén, 23071 Jaén, Spain
e-mail: mjjesus@ujaen.es

Keywords Computer-based education · Data mining · Evolutionary computation · Experimental design · Graphical programming · Java · Knowledge extraction · Machine learning

1 Introduction

Evolutionary Algorithms (EAs) (Eiben and Smith 2003) are optimization algorithms based on natural evolution and genetic processes. Nowadays in Artificial Intelligence (AI), EAs are considered as one of the most successful search techniques for complex problems.

In recent years EAs, particularly Genetic Algorithms (GAs) (Goldberg 1989; Holland 1975), have proved to be an important technique for learning and knowledge extraction. This makes them also a promising tool in Data Mining

V. M. Rivas
e-mail: vrivas@ujaen.es

S. Ventura · C. Romero · J. C. Fernández
Department of Computer Sciences and Numerical Analysis,
University of Córdoba, 14071 Córdoba, Spain
e-mail: sventura@uco.es

C. Romero
e-mail: cromero@uco.es

J. C. Fernández
e-mail: i82fecaj@uco.es

J. M. Garrell
Department of Computer Science,
University Ramon Llull, 08022 Barcelona, Spain
e-mail: josepmg@salle.url.edu

J. Bacardit
Department of Computer Science and Information Technology,
University of Nottingham, NG8 1BB Nottingham, UK
e-mail: jqb@cs.nott.ac.uk

(DM) (Cordón et al. 2001; Freitas 2002; Ghosh and Jain 2005; Grefenstette 1993; Pal and Wang 1996; Wong and Leung 2000). The idea of automatically discovering knowledge from databases is a very attractive and challenging task. Hence, there has been a growing interest in DM in several AI-related areas, including EAs. The main motivation for applying EAs to knowledge extraction tasks is that they are robust and adaptive search methods that perform a global search in place of candidate solutions (for instance, rules or other forms of knowledge representation). The use of EAs in problem solving is a widespread practice. Problems such as image retrieval (Stejić et al. 2007), the learning of controllers in robotics (Mucientes et al. 2006) or the improvement of e-learning systems (Romero et al. 2004) show their suitability as problem solvers in a wide range of scientific fields.

Although EAs are powerful for solving a wide range of scientific problems, their use requires a certain programming expertise along with considerable time and effort to write a computer program for implementing the often sophisticated algorithm according to user needs. This work can be tedious and needs to be done before users can start focusing their attention on the issues that they should be really working on. Given this situation, the aim of this paper is to introduce a non-commercial Java software tool named *KEEL* (Knowledge Extraction based on Evolutionary Learning).[1] This tool empowers the user to analyze the behaviour of evolutionary learning for different kinds of DM problems: regression, classification, unsupervised learning, etc.

This tool can offer several advantages. First of all, it reduces programming work. It includes a library with evolutionary learning algorithms based on different paradigms (Pittsburgh, Michigan and IRL) and simplifies the integration of evolutionary learning algorithms with different preprocessing techniques. It can alleviate researchers from the mere "technical work" of programming and enable them to focus more on the analysis of their new learning models in comparison with the existing ones. Secondly, it extends the range of possible users applying evolutionary learning algorithms. An extensive library of EAs together with easy-to-use software considerably reduce the level of knowledge and experience required by researchers in evolutionary computation. As a result researchers with less knowledge, when using this framework, would be able to apply successfully these algorithms to their problems. Third, due to the use of a strict object-oriented approach for the library and software tool, these can be used on any machine with Java. As a result, any researcher can use KEEL on his machine, independently of the operating system.

This paper is arranged as follows. The next section introduces a study on some non-commercial DM software packages and the main benefits that the KEEL offers with respect to other software tools. Section 3 presents KEEL: its main features and modules. In Sect. 4, two examples are given to illustrate how KEEL should be used. Finally, Sect. 5 points out some conclusions and future work.

2 A study on some non-commercial data mining software

A search on the Internet for DM software reveals the existence of many commercial and non-commercial DM tools and libraries, developed throughout the scientific community. We recommend visiting the KDnuggets software directory[2] and The-Data-Mine site[3] for an overall view of most of them. Although a lot of them are commercially distributed (some of the leading commercial software are mining suites such as SPSS Clementine,[4] Oracle Data Mining[5] and KnowledgeSTUDIO[6]), a few are available as open source-software. Open source tools can play an important role as is pointed out in Sonnenburg et al. (2007).

We can distinguish between libraries whose purpose is to develop new EAs for specific applications and DM suites that incorporate learning algorithms (some of them including evolutionary learning methods) and which in addition provide mechanisms to establish scientific comparisons among them. Over the Internet and in specialized literature we can find a large number of libraries dedicated to evolutionary computation. As generic tools by which it is possible to develop different EAs for many problems we would mention *ECJ* (Luke et al. 2007), *EO* (Keijzer et al. 2001), *Evolvica* (Rummler 2007), *JCLEC* (Ventura et al. 2008) and *Open Beagle* (Gagné and Parizeau 2006). There are also libraries designed for a specific type of EA: genetic algorithms (Chuang 2000), genetic programming Punch and Zongker (1998), memetic algorithms (Krasnogor and Smith 2000), learning classifier systems (Meyer and Hufschlag 2006), evolutionary multiobjective optimization (Tan et al. 2001) or distributed EAs (Tan et al. 2003).

Nowadays, many researchers base their work on DM tools (Rodríguez et al. 2006), or they employ tools specifically designed for an area of DM, such as Wang et al. (2007). We centre our interest on free distributions of software dedicated to the whole range of the DM field. Moreover we are interested in tools where developers, since the source code is available, have the choice of extending their functionality. Probably the most well-known open source DM package is Weka (Witten and Frank 2005), a collection of

[1] http://www.keel.es

[2] http://www.kdnuggets.com/software

[3] http://the-data-mine.com/bin/view/Software

[4] http://www.spss.com/clementine

[5] http://www.oracle.com/technology/products/bi/odm

[6] http://www.angoss.com/products/studio/index.php

Java implementations of Machine Learning (ML) algorithms. Others packages are available as open source software.

The aim of this section is to present a survey of many of such tools, to summarize their strong points and to introduce the reason we have designed KEEL and its benefits.

2.1 Non-commercial suites

In this section we list the open source DM software tools that deserve mention due to their acknowledgement qualities or acceptance.

- *ADaM* (Rushing et al. 2005): This toolkit is packaged as a suite of independent components intended to be used in grid or cluster environments. It provides feature selection capabilities, image processing and data cleaning.
- *D2K (with E2K)* (Llorà 2006): This Data to Knowledge toolkit provides a visual programming environment and a set of templates intended to connect it with other standard packages. It incorporates external packages to perform image and text mining. D2K also offers an external set of evolutionary mechanisms designed for developing basic GAs (E2K).
- *KNIME* (Berthold et al. 2006): This modular environment enables easy integration of new algorithms, data manipulation and visualization methods as models. Compatible with Weka, it also includes statistical methods via the embedded usage of R (R Development Core Team 2005).
- *MiningMart* (Morik and Scholz 2004) is developed with the purpose of re-using best-practice cases of preprocessing high volume data sets. MiningMart is not focused on the whole process of DM but only on one of its steps, the pre-processing chain.
- *Orange* (Demšar and Zupan) is a library of core objects and routines that includes a large variety of standard and not-so-standard ML and DM algorithms, in addition to routines for data input and manipulation. It also includes a scriptable environment for prototyping new algorithms and testing schemes using Python.
- *Tanagra* (Rakotomalala 2005): Tanagra is a DM software for educational and research purposes. It covers several ML schemes, data preparation and experimental analysis.
- *Weka* (Witten and Frank 2005): Weka is the most well-known software tool to perform ML and DM tasks. Its algorithms can either be applied directly to a dataset from its own interface or used in your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Due to its enormous widespread usage, a complete set of extra packages are available for completing its functionalities.
- *RapidMiner* (*formerly YALE*) (Mierswa et al. 2006): It is a free open-source environment for KDD and ML that provides a rich variety of methods which allow the prototyping of new applications and also makes costly re-implementations unnecessary.

All these software tools provide several functionalities, but each one supports them in a different way. In the following subsection we analyze how these software tools tackle a defined set of basic and advanced functionalities.

2.2 Study based on functionality

Having described some of the available DM software tools, we continue with their analysis based on functionality criteria. We do not want to establish a comparison among all software tools or to emphasize the advantages of one over another. Our aim is to point out the main strengths and weakness of each tool in order to compile a set of characteristics in which the existing software tools lack advanced functionality.

With this aim, we have established a set of basic and advanced characteristics that the suites may possess or not. Our objective is to detect the major differences in the software tools and then to categorize KEEL as an alternative to these suites when other research requirements are needed. Table 1 shows a summary of the studied characteristics. All of them have been selected by evaluating all the software tools, tutorials and guidelines for the usage of such suites. The only characteristic that we have added for a different reason is *EAs* integration, given that this is the main motivation for KEEL. We distinguish four levels of support in these characteristics: none (N), basic support (B), intermediate support (I) and advanced support (A). If features do not have intermediate levels of support, the notation used is Yes (Y) for supporting and No (N) for no-supporting.

Selected criteria are briefly explained as follows:

- *Language* is the programming language used in the development of the software. C++ language is less portable with respect to Java.
- *Graphical Interface* includes functionality criteria Which tool can be managed through a handy interface by the user, and how.
  - *Graph representation* indicates that the experiment or knowledge flows are represented by graphs with node-edge connections. This alternative is more interpretable and user-friendly than using a chain of processes or a tree representation of modules.
  - *Data visualization* includes tools for representing the data sets through charts, tables or similar mechanisms.

Table 1 Summary of the characteristics of each DM software tool

Software	Language	Graphical Interface			Input/Output			Pre-processing Variety				Learning Variety				Run Types		Advanced Features			
		Graph representation	Data visualization	Data management	ARFF data format	Other data formats	Data Base connection	Discretization	Feature Selection	Instance Selection	Missing values imputation	Classification	Regression	Clustering	Association Rules	On-line run	Off-line run	Post-processing	Meta-Learning	Statistical tests	EAs
ADaM	C++	N	N	I	Y	N	N	N	A	B	N	I	N	A	B	Y	N	N	N	N	B
D2K	Java	Y	A	I	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	N	N	I
KNIME	Java	Y	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	N	I	B
MiningMart	Java	Y	B	A	N	N	Y	I	A	B	I	B	B	N	N	Y	N	N	N	N	B
Orange	C++	Y	A	A	N	Y	N	A	I	B	B	I	N	I	I	N	Y	N	N	N	N
Tanagra ++	C++	A	A	Y	Y	N	B	A	B	N	A	I	A	A	Y	N	N	I	A	N	
Weka	Java	Y	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	I	N	B
RapidMiner	Java	N	A	A	Y	Y	Y	I	A	B	B	A	A	A	A	Y	N	N	A	B	I

  - *Data management* comprises of a set of toolkits that allow us to perform basic manual operations with the data, such as removing or modifying rows, columns, etc.

- *Input/Output* functionality criteria pointing out the different data formats supported, such as *ARFF* (the Weka standard), *others* (including C4.5 input .names standard (Quinlan 1993), .xls, .csv, XML) and *database connection*. The tool supports this functionality if it can load or save data in these formats or can transform them into a standard one that it uses.
- *Pre-processing Variety*. This comprises of *discretization* (Liu et al. 2002), *feature selection* (Oh et al. 2004), *instance selection* (Wilson and Martinez 2000) and *missing values imputation* (Batista and Monard 2003). The trend of most of the suites is to offer a good feature selection and discretization set of methods, but they overlook specialized methods of missing values imputation and instance selection. Usually, the contributions included are basic modules of replacing or generating null values and methods for sampling the data sets by random (stratified or not) or by value-dependence.
- *Learning Variety* is support over main areas of DM, such as predictive tasks (*classification*, *regression*, *anomaly/deviation detection*), and descriptive tasks (*clustering*, *association rule discovery*, *sequential pattern discovery*) (Tan et al. 2006). Intermediate level is awarded if the tool includes the classical models, and advance level is awarded it the tool contains advanced DM models from these areas.
- *Off/On-line run* of the experiment set up. An On-line run implies that the tool interface and algorithm modules need to be in the same machine and the experiments are completely dependent on the software tool. An off-line run entails the independence of the experiments created with respect to the suite interface, allowing the experiment to be executed in other machines.
- *Advanced Features* includes some of the less common criteria incorporated for extending the functionality of the software tool.

  - *Post-processing*, usually for tuning the model learned by an algorithm.
  - *Meta-learning*, which includes more advanced learning schemes, such as bagging or boosting, or meta learning of the algorithm parameters.
  - *Statistical tests* for establishing comparisons of results. An advanced support of this property requires a complete set of parametric and non-parametric statistical tests; a basic support implies the existence of well-known standard statistical tests (such as t-test).

- *EA* support indicates the integration of EAs into the DM areas that the software tool offers. A basic support of this feature implies the use of genetic algorithms in some techniques (usually, genetic feature selection). To upgrade the level it is necessary to incorporate EAs in learning or meta-learning models.

Analyzing the characteristics presented in Table 1 we can highlight that most of software tools have a none/basic support for two type of pre-processing, statistical tests and EAs. Moreover, the software tools studied usually integrate a representative set of algorithms for each type of learning and pre-processing task. However the experiments are meant to be run in the same environment, which is not practical if the algorithms require high computation times (as with the EAs).

From our point of view users need a software tool where they can analyze the behaviour of evolutionary and non-evolutionary algorithms in each type of learning and pre-processing task, as well as run their experiments in both modes (off-line and on-line). Based on these requirements we have developed the KEEL software tool. In the next section we will describe KEEL in detail.

3 KEEL

KEEL is a software tool that facilitates the analysis of the behaviour of evolutionary learning in the different areas of learning and pre-processing tasks, making the management of these techniques easy for the user. The models correspond with the most well-known and employed models in each methodology, such as evolutionary feature and instance selection (Cano et al. 2003; Llorà and Garrell 2003), evolutionary fuzzy rule learning and Mamdani rule tuning (Alcalá et al. 2006; del Jesus et al. 2004; Otero and Sánchez 2006), genetic artificial neural networks (Martínez-Estudillo et al. 2006; Rivera et al. 2007), Learning Classifier Systems (Bernadó-Mansilla and Ho 2005; Wilson 1995), etc.

The presently available version of KEEL consists of the following function blocks:[7]

- *Data Management*: This part is made up of a set of tools that can be used to build new data, to export and import data in other formats to or from KEEL format, data edition and visualization, to apply transformations and partitioning to data, etc.
- *Design of Experiments (off-line module)*: The aim of this part is the design of the desired experimentation over the selected data sets and providing for many options in different areas: type of validation, type of learning (classification, regression, unsupervised learning), etc...
- *Educational Experiments (on-line module)*: With a similar structure to the previous part, this allows for the design of experiment that can be run step-by-step in order to display the learning process of a certain model by using the software tool for educational purposes.

With all of these function blocks, we can affirm that KEEL can be useful by different types of user, each of whom may expect to find specific features in a DM software.

In the following subsections we describe in detail the user profiles for whom KEEL is intended, its main features and the different integrated function blocks.

3.1 User profiles

KEEL is primarily intended for two categories of users: researchers and students, each of whom have a different set of needs:

- *KEEL as a research tool*: The most common use of this tool for a researcher will be the automation of experiments and the statistical of results. Routinely, an experimental design includes a mix of evolutionary algorithms, statistical and AI-related techniques. Special care was taken to allow a researcher to use KEEL to assess the relevance of his own procedures. Since present standards in ML require heavy computational work, the research tool is not designed to provide a real-time view of the progress of the algorithms but rather to generate a script and be batch-executed in a cluster of computers. The tool allows the researcher to apply the same sequence of pre-processing, experiments and analysis to large batteries of problems and focus his attention on a summary of results.
- *KEEL as an educational tool*: The needs of a student are quite different to those of a researcher. Generally speaking, the aim is no longer that of making statistically sound comparisons between algorithms. There is no need of repeating an experiment a large number of times. If the tool is to be used in class, the execution time must be short and a real-time view of the evolution of the algorithms is needed by the student to learn how to adjust the parameters of the algorithms. In this sense, the educational tool is a simplified version of the research tool, where only the most relevant algorithms are available. Execution is carried out in real time and the user has a visual feedback of the progress of the algorithms, being able to access the final results from the same interface used to design the experimentation in the first place.

[7] http://www.keel.es/software/prototypes/version1.0//ManualKeel.pdf

Each type of user requires the availability of a different set of features in order to be interested in using KEEL. The following subsection will describe the main features of KEEL, covering all the features required by both kinds of user profile.

3.2 Main features

KEEL is a software tool developed to ensemble and use different DM models. We would like to remark that this is the first software toolkit of this type containing a library of evolutionary learning algorithms with open source code in Java. The main features of KEEL are:

- EAs are presented in predicting models, pre-processing (evolutionary feature and instance selection) and post-processing (evolutionary tuning of fuzzy rules).
- Data pre-processing algorithms proposed in specialized literature are included: data transformation, discretization, instance selection and feature selection.
- It contains a statistical library to analyze algorithm results and comprises of a set of statistical tests for analyzing the normality and heteroscedasticity of the results, as well as performing parametric and non-parametric comparisons of the algorithms.
- Some algorithms have been developed using the Java Class Library for Evolutionary Computation (JCLEC) software[8] (Ventura et al. 2008).
- A user-friendly interface is provided, oriented towards the analysis of algorithms.
- The software is designed for experiments containing multiple data sets and algorithms connected among themselves to obtain the desired result. Experiments are independently script-generated from the user interface for an off-line run in the same or other machines.
- KEEL also allows for experiments in on-line mode, intended as an educational support for learning the operation of the algorithms included.
- It contains a Knowledge Extraction Algorithms Library[9] with the incorporation of multiple evolutionary learning algorithms, together with classical learning approaches. The principal families of techniques included are:
  - *Evolutionary rule learning models*. Including different paradigms of evolutionary learning.
  - *Fuzzy systems*. Fuzzy rule learning models with a good trade-off between accuracy and interpretability.
  - *Evolutionary neural networks*. Evolution and pruning in neural networks, product unit neural networks, and radial base models.
  - *Genetic programing*. Evolutionary algorithms that use tree representations for knowledge extraction.
  - *Subgroup discovery*. Algorithms for extracting descriptive rules based on patterns subgroup discovery.
  - *Data reduction (instance and feature selection and discretization)*. EAs for data reduction.

KEEL integrates the library of algorithms in each of its the function blocks. We have briefly presented its function blocks above but in the following subsections, we will describe the possibilities that KEEL offers in relation to data management, off-line experiment design and on-line educational design.

3.3 Data management

The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed, or made more accessible (Pyle 1999). Data preparation comprises those techniques concerned with analyzing raw data so as to yield quality data, mainly including data collecting, data integration, data transformation, data cleaning, data reduction and data discretization (Zhang et al. 2003). Data preparation can be even more time consuming than data mining, and can present equal challenges to data mining. Its importance lies in that the real-world data is impure (incomplete, noisy and inconsistent) and high-performance mining systems require quality data (the removal of anomalies or duplications). Quality data yields high-quality patterns (to recover missing data, purify data and resolve conflicts).

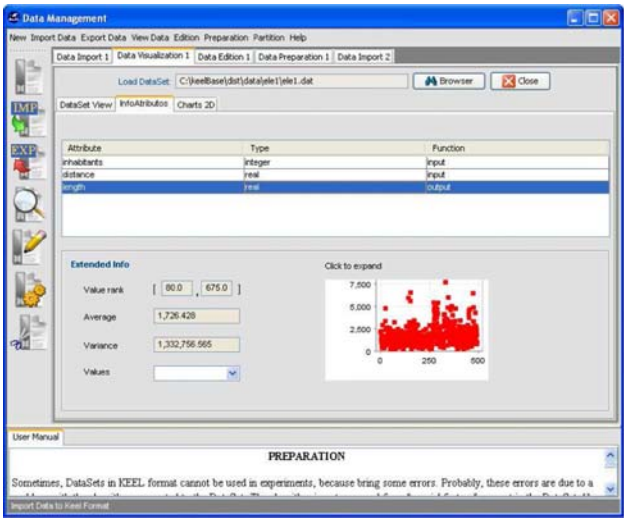
The *Data Management* module integrated in KEEL allows us to perform the data preparation stage independently of the remaining of the DM process itself. This module is focused on the group of users denoted as domain experts. They are familiar with their data, they know the processes that produce the data and they are interested in reviewing those to improve upon or analyze them. On the other hand, domain users are those whose interest lies in applying processes to their own data and they usually are not experts in DM.

Figure 1 shows an example window of the *Data Management* module in the section of *Data Visualization*. The module has seven sections, each of which is accessible through the buttons on the left side of the window. In the following, we will briefly describe them:

- *Creation of a new data set*: This option allows us to generate a new data set compatible with the other KEEL modules.
- *Import data to KEEL format*: Since KEEL works with a specific data format (alike the ARFF format) in all its modules, this section allows us to convert various data

[8] http://jclec.sourceforge.net/

[9] http://www.keel.es/algorithms.php



Load DataSet: C:\keelBase\dist\data\ele1\ele1.dat

Attribute	Type	Function
inhabitants	integer	input
distance	real	input
length	real	output

Extended Info

Value rank [80.0 , 675.0]

Average 1,726.428

Variance 1,332,756.565

Values

PREPARATION

Sometimes, DataSets in KEEL format cannot be used in experiments, because bring some errors. Probably, these errors are due to a

Fig. 1 Data Management

formats to KEEL format, such as CSV, XML, ARFF, extracting data from data bases, etc.

- *Export data from KEEL format*: This is the opposite option to the previous one. It converts the data handled by KEEL procedures in other external formats to establish compatibility with other software tools.
- *Visualization of data*: This option is used to represent and visualize the data. With it, we can see a graphical distribution of each attribute and comparisons between two attributes.
- *Edition of data*: This area is dedicated to managing the data manually. The data set, once loaded, can be edited by terms of modifying values, adding or removing rows and columns, etc.
- *Data Partition*: This zone allows us to make the partitions of data needed by the experiment modules to validate results. It supports k-fold cross validation, 5×2 cross validation and hold-out validation with stratified partition.
- *Data Preparation*: This section allows us to perform automatic data preparation for DM, including cleaning, transformation and reduction of data. All techniques integrated in this section are also available in the experiments-related modules.

3.4 Design of experiments: off-line module

In the last few years, a large number of DM software tools have been developed for research purposes. Some of them are libraries that allow reductions in programming work when developing new algorithms: ECJ (Luke et al. 2007), JCLEC (Ventura et al. 2008), learning classifier systems (Meyer and Hufschlag 2006), etc. Others are DM suites that incorporate learning algorithms (some of them may use EAs for this task) and provide a mechanism to establish comparisons among them. Some examples are Weka (Witten and Frank 2005), D2K (Llorà 2006), etc.

This module is a Graphical User Interface (GUI) that allows the design of experiments for solving various problems of regression, classification and unsupervised learning. Having designed the experiments, it generates the directory structure and files required for running them in any local machine with Java (see Fig. 2).

The experiments are graphically modeled, based on data flow and represented by graphs with node-edge connections. To design an experiment, we have first to indicate the type of validation (k-fold cross validation (Kohavi 1995) or 5×2 cross validation (Dietterich 1998) and the type of learning (regression, classification or unsupervised) to be used. Then, we have to select the data sources, drag the selected methods into the workspace and connect methods and datasets, combining the evolutionary learning algorithms with different pre-processing and post-processing techniques, if needed. Finally, we can add statistical tests to achieve a complete analysis of the methods being studied, and a report box to obtain a summary of the results. Notice that each component of the experiment is configured in separate dialogues that can be opened by double-clicking the respective node. Figure 3 shows an example of an experiment following the MOGUL methodology (Cordón et al. 1998) and using a report box to obtain a summary of the results. The configuration window of the MOGUL method is also shown in this figure.

When the experiment has been designed, the user can choose either to save the design in a XML file or to obtain a zip file. If the user chooses a zip file, then the system will generate the file with the directory structure and required files for running the designed experiment in any local machine with Java. This directory structure contains the data sources,

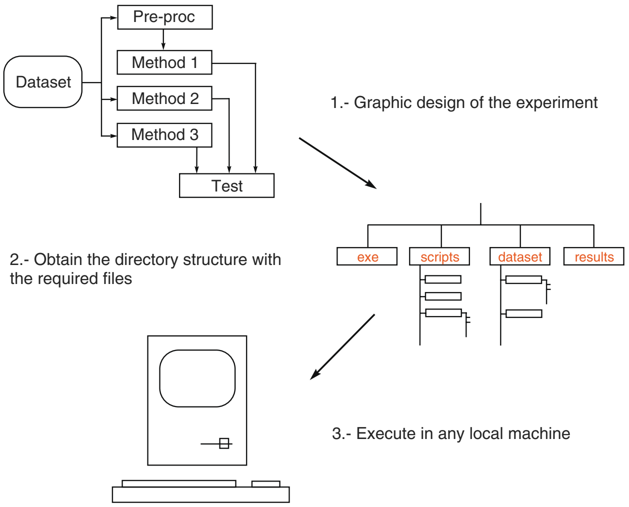
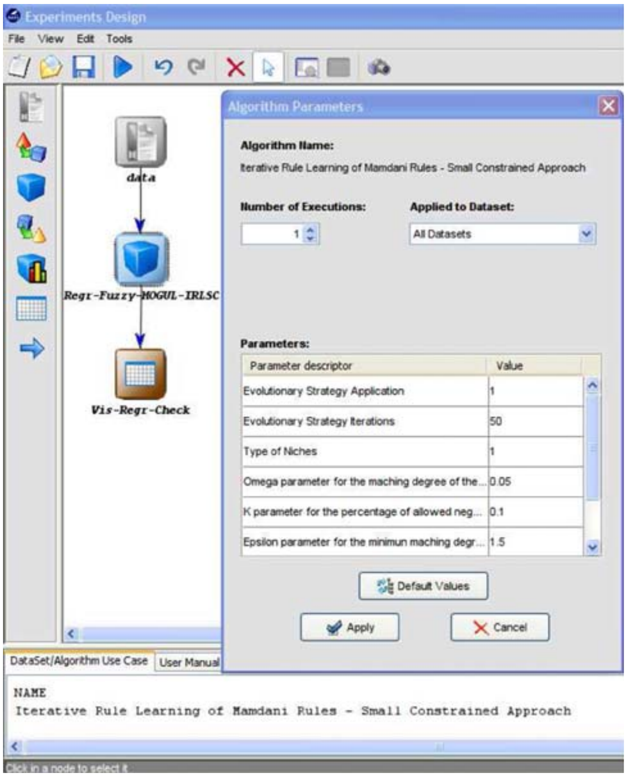

Pre-proc
Dataset
Method 1
Method 2
Method 3
Test
1.- Graphic design of the experiment
2.- Obtain the directory structure with the required files
exe
scripts
dataset
results
3.- Execute in any local machine

Fig. 2 Design of experiments



Algorithm Parameters

Algorithm Name:
Iterative Rule Learning of Mamdani Rules - Small Constrained Approach

Number of Executions: 1
Applied to Dataset: All Datasets

Parameters:

Parameter descriptor	Value
Evolutionary Strategy Application	1
Evolutionary Strategy Iterations	50
Type of Niches	1
Omega parameter for the maching degree of the...	0.05
K parameter for the percentage of allowed neg...	0.1
Epsilon parameter for the minimun maching degr...	1.5

```
NAME
Iterative Rule Learning of Mamdani Rules - Small Constrained Approach
```

Fig. 3 Example of an experiment and the configuration window of a method

the jar files of the algorithms, the configuration files in XML format, a script file with all the indicated algorithms in XML format, and a Java tool, named *RunKeel*, to run the experiment. *RunKeel* can be seen as a simple EA scripting environment that reads the script file in XML format, runs all the indicated algorithms and saves the results in one or several report files.

Obviously, this kind of interface is ideal for experts of specific areas who, knowing the methodologies and methods used in their particular area of interest, intend to develop a new method and would like to compare it with the well-known methods available in KEEL.

3.5 Computer-based education: on-line module

There is a variety of terms used to describe the use of computers in education (Ortega and Bravo 2000). Computer-assisted instruction (CAI), computer-based education (CBE) and computer-based instruction (CBI) are the broadest terms and can refer to virtually any kind of computer use in educational environments. These terms may refer either to stand-alone computer learning activities or to computer activities which reinforce material introduced and taught by teachers.

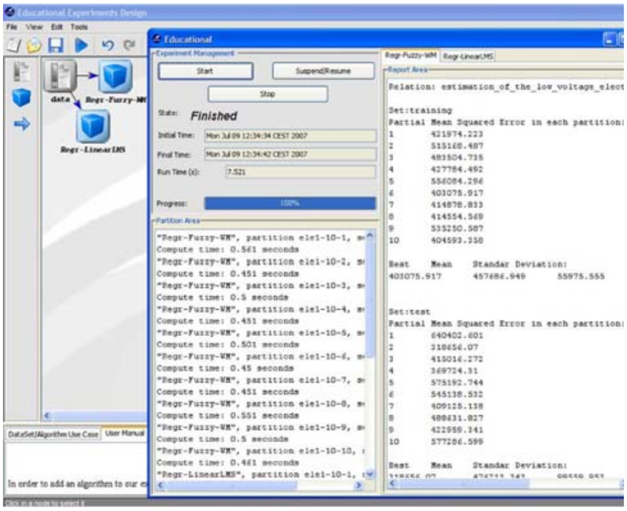
Most of the software developed in DM and evolutionary computation domain is designed for research purposes (libraries, algorithms, specific applications, etc.). But there is some free software that are designed not only for research but also for educational purposes. These systems are easy to use due to the fact that they provide a GUI to assist user interaction with the system in all the tasks (selecting data, choosing parameters, running algorithms, visualize the results, etc.). Some examples of open source DM systems are Weka (Witten and Frank 2005), RapidMiner (Mierswa et al. 2006) and Tanagra (Rakotomalala 2005).

This module is a GUI that allows the user to design an experiment (with one or more algorithms), run it and visualize the results on-line. The idea is to use this part of KEEL as a guideline to demonstrate the learning process of a certain model. This module has a similar structure as the previous one but includes only those algorithms and options that are suitable for academic purposes.

When an experiment is designed the user can choose either to save the experiment in a XML file or to run it. If the user chooses to run it, then the system will show an auxiliary window to manage and visualize the execution of each algorithm. When the run finishes, this window will show the results obtained for each algorithm in separate tags, showing for example the confusion matrices for classification or the mean square errors for regression problems (see Fig. 4).

4 Case studies

This section presents two case studies as examples of the functionality and process of creating an experiment in the KEEL software tool. The first study is focused on the development of a comparison of some algorithms and a subsequent analysis of the results using the off-line module. The



State: *Finished*
Initial Time: Mon Jul 09 12:34:34 CEST 2007
Final Time: Mon Jul 09 12:34:42 CEST 2007
Run Time (s): 7.521

```
"Regr-Fuzzy-WM", partition ele1-10-1, m
Compute time: 0.561 seconds
"Regr-Fuzzy-WM", partition ele1-10-2, m
Compute time: 0.451 seconds
"Regr-Fuzzy-WM", partition ele1-10-3, m
Compute time: 0.5 seconds
"Regr-Fuzzy-WM", partition ele1-10-4, m
Compute time: 0.451 seconds
"Regr-Fuzzy-WM", partition ele1-10-5, m
Compute time: 0.501 seconds
"Regr-Fuzzy-WM", partition ele1-10-6, m
Compute time: 0.45 seconds
"Regr-Fuzzy-WM", partition ele1-10-7, m
Compute time: 0.451 seconds
"Regr-Fuzzy-WM", partition ele1-10-8, m
Compute time: 0.551 seconds
"Regr-Fuzzy-WM", partition ele1-10-9, m
Compute time: 0.5 seconds
"Regr-Fuzzy-WM", partition ele1-10-10, :
Compute time: 0.461 seconds
"Regr-LinearLMS", partition ele1-10-1, :
```

```
Relation: estimation_of_the_low_voltage_elect

Set:training
Partial Mean Squared Error in each partition:
1       421974.223
2       515168.487
3       483504.735
4       427784.492
5       556084.296
6       403075.917
7       414878.833
8       414554.569
9       535250.587
10      404593.350

Best    Mean    Standar Deviation:
403075.917      457686.949      55975.555


Set:test
Partial Mean Squared Error in each partition:
1       640402.601
2       318656.07
3       415016.272
4       369724.31
5       575192.744
6       545138.532
7       409125.138
8       488631.827
9       422959.341
10      577286.599

Best    Mean    Standar Deviation:
```

Fig. 4 Auxiliary window of an experiment with two algorithms

second example is a presentation of the educational on-line module.

4.1 Off-line case study

Our purpose in this example is to make a comparison of three methods that belong to different ML techniques and use EAs in the learning task. The experiment graph is represented in Fig. 5. In this example, we have used a k-Nearest Neighbour classifier with a previous pre-processing stage of prototype selection guided by a CHC model (IS-CHC + Clas-KNN) (Cano et al. 2003). We have also used a XCS classifier (Wilson 1995) and an Evolutionary Product Unit based Neural Networks (NNEP) (Martínez-Estudillo et al. 2006).

By clicking the *Experiment* option in the main menu of the KEEL software tool, we define the experiment as a *Classification* problem and we use a *10-fold cross validation* procedure to analyze the results. Next, the first step of the experiment graph set-up is to choose the data sets to be used. This example uses only the *Iris* data set, but more that one data set may be chosen at the same time.

The graph in Fig. 5 represents the flow of data and results from the algorithms and procedures. A node can represent an initial data flow (data set), a pre-process/post-process algorithm, a learning method, test, or a visualization of results module. They can be easily distinguished according to the color of the node. All their parameters can be adjusted by clicking twice on the node. Logically, directed edges connecting two nodes imply a relation between them (data or results interchange). When the data is interchanged, the flow includes pairs of train-test data sets in classification and regression problems. Thus, the graph in this specific example describes a flow of data from the *Iris* data set to three nodes, two of which are learning methods. Above them, the flow of data is the input of a pre-process method, which operation consists of reducing the training data by removing instances. The resultant subset is used subsequently as a reference set for the k-NN classifier. XCS and NNEP will use the full training data for learning the model.

After the models are trained, the instances of the data set are classified according to the training and test files. These results are the inputs for the visualization and test modules. The module *Vis-Clas-Tabular* receives these results as inputs and generates output files with several performance metrics computed from them, such as confusion matrices for each method, accuracy and error percentages for each method, fold and class, and a final summary of results. Figure 5 also shows another type of results flow, interconnecting each possible pair of methods with a test module. In this case, the test module used is the signed-rank Wilcoxon non-parametrical procedure *Stat-Clas-WilcoxonSR* for comparing two samples of results. The experiment establishes a pair-wise statistical comparison of the three methods.

Once the graph has been defined, we can set up the associated experiment and save it as zip file for an off-line run. Following the structure of directories shown in Fig. 2, the experiment is set up as a set of XML scripts and a jar program for running it. Within the *results* directory, there will be directories used for housing the results of the methods during the run. If the method is a learning method, its associated directory will house the learned model. In case of a test/visualization procedure, its directory will house the results files. Once the experiment has been run we can find the result file of the confusion matrix (see Fig. 6 for the confusion matrix of the IS-CHC + Clas-KNN classifier) or the one associated with a Wilcoxon comparison (Fig. 7).

With a simple design of a logical flow of data by means of a graph representation, a user can set up an experiment involving several data sets, interconnect pre-processing tasks with learning tasks, integrate and configure powerful learning models with classical ones, compile the obtained results establishing statistical comparisons of them, and finally run the entire process in another machine, the only requirement being that Java is installed.

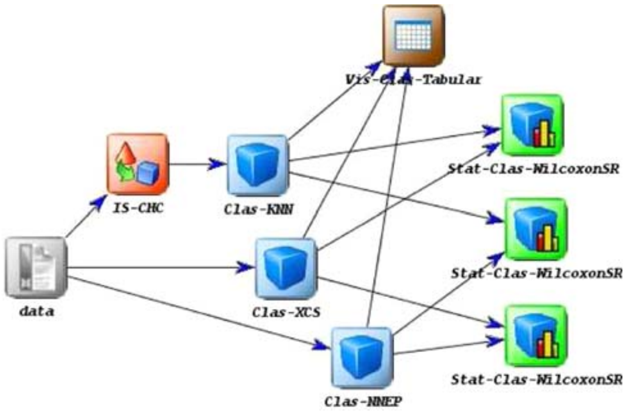
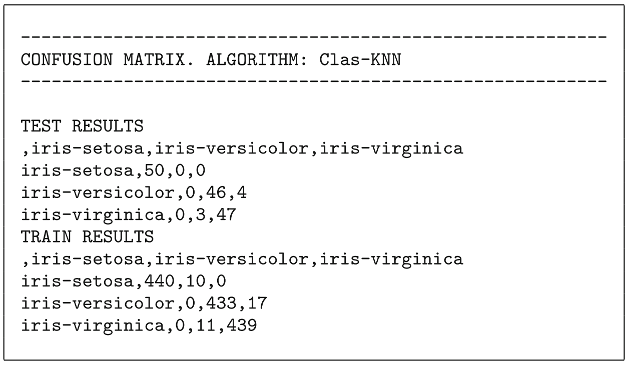


Fig. 5 Experiment graph of the off-line example



```
------------------------------------------------------------
CONFUSION MATRIX. ALGORITHM: Clas-KNN
------------------------------------------------------------

TEST RESULTS
,iris-setosa,iris-versicolor,iris-virginica
iris-setosa,50,0,0
iris-versicolor,0,46,4
iris-virginica,0,3,47
TRAIN RESULTS
,iris-setosa,iris-versicolor,iris-virginica
iris-setosa,440,10,0
iris-versicolor,0,433,17
iris-virginica,0,11,439
```

Fig. 6 Confusion Matrix obtained for IS-CHC + Clas-KNN

```
Wilcoxon signed rank test, Classification
Classification error in each fold:
Algorithm = 0
Fold 0 : 0.06666666666666667
Fold 1 : 0.06666666666666667
Fold 2 : 0.06666666666666667
Fold 3 : 0.06666666666666667
Fold 4 : 0.06666666666666667
Fold 5 : 0.0
Fold 6 : 0.0
Fold 7 : 0.13333333333333333
Fold 8 : 0.06666666666666667
Fold 9 : 0.13333333333333333
Algorithm = 1
Fold 0 : 0.0
Fold 1 : 0.0
Fold 2 : 0.0
Fold 3 : 0.06666666666666667
Fold 4 : 0.0
Fold 5 : 0.0
Fold 6 : 0.06666666666666667
Fold 7 : 0.13333333333333333
Fold 8 : 0.0
Fold 9 : 0.06666666666666667
Null hypothesis, true difference in means is equal to 0
Output=0: There is no evidence against H0
p-value: 0.09934224785065712
```

Fig. 7 Results for signed-rank Wilcoxon test comparing XCS with NNEP

4.2 On-line case study

An example of an educational application of KEEL is shown in Fig. 8. Our purpose is to observe the learning process of a regression algorithm using fuzzy rule learning (Wang and Mendel 1992) over an *electrical energy distribution problem* (Cordón et al. 1999) by using five labels per variable.

The run of the experiment takes place in a new window (see Fig. 9). The user/teacher can start, suspend/pause and stop the experiment at any moment in order to see step by step partial reports of the execution. Information provided after the run finishes is the run-time, accuracy in each fold, average accuracy, and a brief description of functions used for evaluating accuracy of the solutions.

This GUI is ideal for students who can interpret the results and learn to change parameters to improve results. In this example, they can prove for themselves that Wand and Mendel's ad-hoc method (Wang and Mendel 1992) is very fast but its results far from accurate.

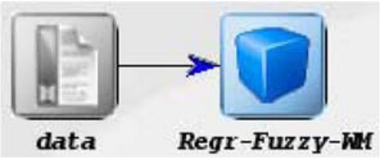
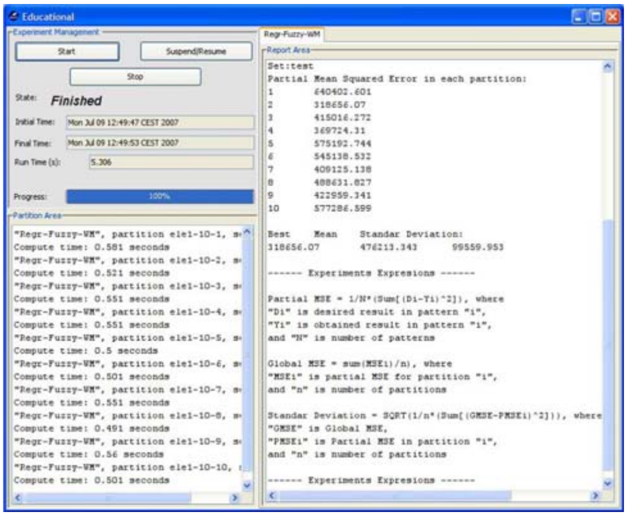


Fig. 8 Experiment graph of the on-line example



State: Finished

Initial Time: Mon Jul 09 12:49:47 CEST 2007

Final Time: Mon Jul 09 12:49:53 CEST 2007

Run Time (s): 5.306

```
"Regr-Fuzzy-WM", partition ele1-10-1, s
Compute time: 0.581 seconds
"Regr-Fuzzy-WM", partition ele1-10-2, s
Compute time: 0.521 seconds
"Regr-Fuzzy-WM", partition ele1-10-3, s
Compute time: 0.551 seconds
"Regr-Fuzzy-WM", partition ele1-10-4, s
Compute time: 0.551 seconds
"Regr-Fuzzy-WM", partition ele1-10-5, s
Compute time: 0.5 seconds
"Regr-Fuzzy-WM", partition ele1-10-6, s
Compute time: 0.501 seconds
"Regr-Fuzzy-WM", partition ele1-10-7, s
Compute time: 0.551 seconds
"Regr-Fuzzy-WM", partition ele1-10-8, s
Compute time: 0.491 seconds
"Regr-Fuzzy-WM", partition ele1-10-9, s
Compute time: 0.56 seconds
"Regr-Fuzzy-WM", partition ele1-10-10,
Compute time: 0.501 seconds
```

```
Set:test
Partial Mean Squared Error in each partition:
1       640402.601
2       318656.07
3       415016.272
4       369724.31
5       575192.744
6       545138.532
7       409125.138
8       488631.827
9       422959.341
10      577286.599

Best    Mean    Standar Deviation:
318656.07       476213.343      99559.953

------ Experiments Expresions ------

Partial MSE = 1/N*(Sum[(Di-Yi)^2]), where
"Di" is desired result in pattern "i",
"Yi" is obtained result in pattern "i",
and "N" is number of patterns

Global MSE = sum(MSEi)/n), where
"MSEi" is partial MSE for partition "i",
and "n" is number of partitions

Standar Deviation = SQRT(1/n*(Sum[(GMSE-PMSEi)^2])), where
"GMSE" is Global MSE,
"PMSEi" is Partial MSE in partition "i",
and "n" is number of partitions

------ Experiments Expresions ------
```

Fig. 9 Window of results obtained in the experiment

5 Conclusion and future work

In this paper, we have described a non-commercial Java software tool, named KEEL, that provides a software tool for the analysis of evolutionary learning methods applied to Data Mining problems. This tool relieves researchers of much technical work and allows them to focus on the analysis of their new learning models in comparison with the existing ones. Moreover, the tool enables researchers with little knowledge of evolutionary computation methods to apply evolutionary learning algorithms to their work.

We have shown the main features of this software tool and we have distinguished three main parts: a module for data management, a module for designing experiments with evolutionary learning algorithms, and a module educational with goals. We have also shown two case studies to illustrate functionalities and the experiment set up processes.

The KEEL software tool is being continuously updated and improved. At the moment, we are developing a new set of evolutionary learning algorithms and a test tool that will allow us to apply parametric and non-parametric tests on any set of data. We are also developing data visualization tools for the on-line and off-line modules, as well as a graphical tool to run in a distributed environment the experiments designed with the off-line module. Finally, we are also working on the development of a data set repository that includes the data set partitions and algorithm results on these data sets, the *KEEL-dataset*.[10]

[10] http://www.keel.es/datasets.php

References

Alcalá R, Alcala-Fdez J, Casillas J, Cordón O, Herrera F (2006) Hybrid learning models to get the interpretabilityaccuracy trade-off in fuzzy modeling. Soft Comput 10(9):717–734

Batista GE, Monard MC (2003) An analysis of four missing data treatment methods for supervised learning. Appl Artif Intell 17:519–533

Bernadó-Mansilla E, Ho TK (2005) Domain of competence of XCS classifier system in complexity measurement space. IEEE Trans Evol Comput 9(1):82–104

Berthold MR, Cebron N, Dill F, Di Fatta G, Gabriel TR, Georg F, Meinl T, Ohl P (2006) KNIME: The Konstanz Information Miner, In: Proceedings of the 4th annual industrial simulation conference, Workshop on multi-agent systems and simulations, Palermo

Cano JR, Herrera F, Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. IEEE Trans Evol Comput 7(6):561–575

Cordón O, del Jesus MJ, Herrera F, Lozano M (1999) MOGUL: a methodology to obtain genetic fuzzy rule-based systems under the iterative rule learning approach. Int J Intell Syst 14(9):1123–1153

Cordón O, Herrera F, Sánchez L (1999) Solving electrical distribution problems using hybrid evolutionary data analysis techniques. Appl Intell 10:5–24

Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases. World Scientific, Singapore, p 488

Chuang AS (2000) An extendible genetic algorithm framework for problem solving in a common environment. IEEE Trans Power Syst 15(1):269–275

del Jesus MJ, Hoffmann F, Navascues LJ, Sánchez L (2004) Induction of Fuzzy-Rule-Based Classifiers with Evolutionary Boosting Algorithms. IEEE Trans Fuzzy Syst 12(3):296–308

Demšar J, Zupan B Orange: From experimental machine learning to interactive data mining, White Paper (http://www.ailab.si/orange). Faculty of Computer and Information Science, University of Ljubljana

Dietterich TG (1998) Approximate Statistica Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation 10(7):1895–1924

Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, Berlin, p 299

Freitas AA (2002) Data mining and knowledge discovery with evolutionary algorithms. Springer, Berlin, p 264

Gagné C, Parizeau M (2006) Genericity in evolutionary computation sofyware tools: principles and case-study Int J Artif Intell Tools 15(2):173–194

Ghosh A, Jain LC (2005) Evolutionary Computation in Data Mining. Springer, New York pp 264

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, New York, pp 372

Grefenstette JJ (1993) Genetic Algorithms for Machine Learning. Kluwer, Norwell, p 176

Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, London, p 228

Keijzer M, Merelo JJ, Romero G, Schoenauer M (2001) Evolving objects: A general purpose evolutionary computation library. In: Collet P, Fonlupt C, Hao JK, Lutton E, Schoenauer M (eds) Artificial evolution: selected papers from the 5th european conference on artificial evolution, London, UK, pp 231–244

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence 2(12):1137–1143

Krasnogor N, Smith J (2000) MAFRA: A Java memetic algorithms framework. In: Proceedings of the Genetic and Evolutionary Computation Workshops. Las Vegas, Nevada, USA, pp 125–131

Llorà X (2006) E2K: Evolution to knowledge. SIGEVOlution 1(3):10–16

Llorà X, Garrell JM (2003) Prototype induction and attribute selection via evolutionary algorithms. Int Data Anal 7(3):193–208

Liu H, Hussain F, Lim C, Dash M (2002) Discretization: an enabling technique. Data Min Knowl Discov 6(4):393–423

Luke S, Panait L, Balan G, Paus S, Skolicki Z, Bassett J, Hubley R, Chircop A (2007) ECJ: A Java based evolutionary computation research system. http://cs.gmu.edu/~eclab/projects/ecj

Martínez-Estudillo A, Martínez-Estudillo F, Hervás-Martínez C, García-Pedrajas N (2006) Evolutionary product unit based neural networks for regression. Neural Netw 19:477–486

Meyer M, Hufschlag K (2006) A generic approach to an object-oriented learning classifier system library. Journal of Artificial Societies and Social Simulation 9:3 http://jasss.soc.surrey.ac.uk/9/3/9.html

Mierswa I, Wurst M, Klinkenberg R, Scholz M, Euler T (2006) YALE: Rapid Prototyping for Complex Data Mining Tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 1–6

Morik K, Scholz M (2004) The MiningMart Approach to Knowledge Discovery in Databases. In: Zhong N, Liu J (eds) Intelligent Technologies for Information Analysis. Springer, Heidelberg, pp 47–65

Mucientes M, Moreno DL, Bugarín A, Barro S (2006) Evolutionary learning of a fuzzy controller for wallfollowing behavior in mobile robotics. Soft Comput 10(10):881–889

Oh IS, Lee JS, Moon BR (2004) Hybrid genetic algorithms for feature selection. IEEE Trans Pattern Anal Mach Intell 26(11):1424–1437

Ortega M, Bravo J (2000) Computers and education in the 21st century. Kluwer, Norwell, p 266

Otero J, Sánchez L (2006) Induction of descriptive fuzzy classifiers with the Logitboost algorithm. Soft Comput 10(9):825–835

Pal SK, Wang PP (1996) Genetic algorithms for pattern recognition. CRC Press, Boca Raton,p 336

Punch B, Zongker D (1998) lib-gp 1.1 beta. http://garage.cse.msu.edu/software/lil-gp

Pyle D (1999) Data preparation for data mining. Morgan Kaufmann, San Mateo, p 540

Quinlan JR (1993) C4.5: programs for machine learning. Morgan Kaufmann, San Mateo, p 316

R Development Core Team (2005) R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria http://www.R-project.org

Rakotomalala R (2005) TANAGRA: un logiciel gratuit pour l'enseignement et la recherche. In: Proceedings of the 5th Journées d'Extraction et Gestion des Connaissances 2:697–702

Rivera AJ, Rojas I, Ortega J, del Jesus MJ (2007) A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. Soft Comput 11(7):655–668

Rodríguez JJ, Kuncheva LI, Alonso CJ (2006) Rotation forest: a new classifier ensemble method. IEEE Trans Pattern Anal Mach Intell 28(10):1619–1630

Romero C, Ventura S , de Bra P (2004) Knowledge discovery with genetic programming for providing feedback to courseware author, user modeling and user-adapted interaction. J Personal Res 14(5):425–465

Rummler A (2007) Evolvica: a Java framework for evolutionary algorithms. http://www.evolvica.org

Rushing J, Ramachandran R, Nair U, Graves S, Welch R, Lin H (2005) ADaM: a data mining toolkit for scientists and engineers. Comput Geosci 31(5):607–618

Sonnenburg S, Braun ML, Ong ChS, Bengio S, Bottou L, Holmes G, LeCun Y, Müller K-R, Pereira F, Rasmussen CE, Rätsch G, Schölkopf B, Smola A, Vincent P, Weston J, Williamson RC (2007) The need for open source software in machine learning. J Mach Learn Res 8:2443–2466

Stejić Z, Takama Y, Hirota K (2007) Variants of evolutionary learning for interactive image retrieval. Soft Comput 11(7):669–678

Tan JC, Lee TH, Khoo D, Khor EF (2001) A multiobjective evolutionary algorithm toolbox for computer-aided multiobjective optimization. IEEE Trans Syst Man Cybern B Cybern 31(4):537–556

Tan JC, Tay A, Cai J (2003) Design and implementation of a distributed evolutionary computing software. IEEE Trans Syst Man Cybern B Cybern 33(3):325–338

Tan PN, Steinbach M, Kumar V (2006) Introduction to Data Mining. Addison-Wesley, Reading, p 769

Ventura S, Romero C, Zafra A, Delgado JA, Hervás C (2008) JCLEC: a java framework for evolutionary computation. Soft Comput 12(4):381–392

Wang LX, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 22(6):1414–1427

Wang X, Nauck DD, Spott M, Kruse R (2007) Intelligent data analysis with fuzzy decision trees. Soft Comput 11(5):439–457

Wilson SW (1995) Classifier fitness based on accuracy. Evol Comput 3(2):149–175

Wilson DR, Martinez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38:257–268

Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco, p 525. http://www.cs.waikato.ac.nz/ml/weka/index.html

Wong ML, Leung KS (2000) Data mining using grammar based genetic programming and applications. Kluwer, Norwell, p 232

Zhang S, Zhang C, Yang Q (2003) Data preparation for data mining. Appl Artif Intell 17:375–381