Assessing Evolutionary Algorithms to Data Mining Problems

Jesús Alcalá
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
Introduction

• Data Mining (DM) is the process for automatic discovery of high level knowledge by obtaining information from real world, large and complex data sets.

• EAs have proved to be an important technique for learning and knowledge extraction. This makes them a promising tool in Data Mining.

• Evolutionary algorithms requires a certain programming expertise along with considerable time and effort to write a computer program for implementing algorithms that often are sophisticated.
Introduction

• In the last few years, many software tools have been developed to reduce this task.

• **Open source tools can play an important role** as is pointed out in:

Introduction

KEEL (Knowledge Extraction based on Evolutionary Learning) is an open source (GPLv3) Java software tool which empowers the user to assess the behavior of evolutionary learning and Soft Computing based techniques for different kinds of DM problems: regression, classification, clustering, pattern mining and so on.

http://www.keel.es


Granada, October 22nd, 2010
Introduction

This tool can offer several advantages:

• It includes a big library with evolutionary learning algorithms based on different paradigms (Pittsburgh, Michigan, IRL and GCCL) and simplifies their integration with different pre-processing techniques.

• It extends the range of possible users applying evolutionary learning algorithms.

• KEEL can be used on any machine with Java.
Introduction

KEEL is being developed under the Spanish National Projects TIC2002-04036-C05, TIN2005-08386-C05 and TIN2008-06681-C06 with the collaboration of the six following Spanish Research Groups:
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
KEEL

• KEEL is a software tool to assess EAs for DM problems including regression, classification, clustering, pattern mining and so on.

• KEEL allows us to perform a complete analysis of any learning model in comparison to existing ones, including a statistical test module for comparison.

• Moreover, KEEL has been designed with a double goal: research and educational.
KEEL - Main Features

- EAs are presented in predicting models, pre-processing and postprocessing.
- It includes data pre-processing algorithms proposed in specialized literature: data transformation, discretization, instance selection and feature selection.
- It contains a statistical library for analyzing results
- It provides a user-friendly graphical interface.
- It contains a Knowledge Extraction Algorithms Library. The main employment lines are:
  - Different evolutionary rule learning models have been implemented
  - Fuzzy rule learning models with a good trade-off between accuracy and interpretability.
  - Genetic Programming: Evolutionary algorithms that use tree representations for extracting knowledge.
  - Algorithms for extracting descriptive rules based on patterns subgroup discovery have been integrated.
  - Data reduction (training set selection, feature selection and discretization). EAs for data reduction have been included.
KEEL - Blocks

It is integrated by three main blocks:

- Data Management.
- Design of Experiments (off-line module).
- Educational Experiments (on-line module).

And two specific blocks:

- Imbalanced Experiments.
- Statistical Tests.
KEEL - Blocks

Data Management

- Import Data
- Export Data
- Visualize Data
- Edit Data
- Partition Data
KEEL - Blocks

Design of Experiments

• It is a Graphical User Interface that allows the design of experiments for solving different machine learning problems.

• Once the experiment is designed, it generates the directory structure and files required for running them in any local machine with Java.
KEEL - Blocks

Design of Experiments

- The experiments are **graphically modeled**. They represent a multiple connection among data, algorithms and analysis/visualization modules.

- Aspects such as type of learning, validation, number of runs and algorithm’s parameters can be easily configured.

- Once the experiment is created, KEEL generates a **script-based program** which can be run in any machine with JAVA Virtual Machine installed in it.
KEEL - Blocks

Educational Experiments

- Similar structure to the design of experiments
- This allows for the design of experiments 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.
- Results and analysis are shown in on-line mode.
KEEL - Blocks

Statistical Test

KEEL is one of the fewest Data Mining software tools that provides to the researcher a complete set of statistical procedures for pairwise and multiple comparisons. Inside the KEEL environment, several parametric and nonparametric procedures have been coded, which should help to contrast the results obtained in any experiment performed with the software tool.
KEEL - Blocks

Imbalanced Experiments

The aim of this part is the design of the desired experimentation over the selected imbalanced data sets. These experiments are created for 5cfv datasets and include specific algorithms for imbalanced data and general classification algorithms.
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
Why open source?

- KEEL has been developed with the idea of being easily extended with new algorithms.

- Our aim in this work is to offer the possibility for researchers to integrate their own approaches in KEEL introducing some basic guidelines that the developer may take into account for managing the specific constraints of KEEL.

- A source code template have been made to manage all the restrictions of the KEEL software.
Integration of New Algorithms

List of details to take into account before codifying a method for KEEL:

1. The programming language used is **Java**.

2. The **parameters** are read from a single file, which includes:
   - The **name** of the algorithm
   - The **path** of the input and output files
   - **List of parameter’s values** for the algorithm.
Integration of New Algorithms

3. The input data-sets follow the KEEL format that extends the ARFF format by completing the header with more information about the attributes.

4. The output format consists of:
   - A header, which follows the same scheme as the input data
   - Two columns with the output values for each example separated with a white space

<table>
<thead>
<tr>
<th>Examples</th>
<th>Predicted Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Output</td>
</tr>
<tr>
<td>1.9, 3.5</td>
<td>Red Yellow</td>
</tr>
<tr>
<td>0.5, 9.1</td>
<td>Blue Blue</td>
</tr>
</tbody>
</table>

Integration of New Algorithms

- The KEEL development team have created a simple template that manages all these features.

- KEEL template includes four classes:
  - **Main**: This class contains the main instructions for launching the algorithm.
  - **ParseParameters**: This class manages all the parameters.
  - **myDataset**: This class is an interface between the classes of the API dataset and the algorithm.
  - **Algorithm**: This class is devoted to store the main variables of the algorithm and to call the different procedures for the learning stage

http://www.keel.es/software/KEEL_template.zip
Example

- We have selected one classical and simple method, the Chi et al.'s rule learning procedure.

- Neither the Main nor ParseParameters nor myDataset classes need to be modified.

- We need to only focus our effort on the Algorithm class.
Example

1. **Store all the parameter’s values within the constructor of the algorithm**

```java
public Fuzzy_Chi(parseParameters parameters) {
    try {
        System.out.println("Reading the training set: " + parameters.getTrainingInputFile());
        train.readClassificationSet(parameters.getTrainingInputFile(), true);
        System.out.println("Reading the validation set: " + parameters.getValidationInputFile());
        val.readClassificationSet(parameters.getValidationInputFile(), false);
        System.out.println("Reading the test set: " + parameters.getTestInputFile());
        test.readClassificationSet(parameters.getTestInputFile(), false);
    } catch (IOException e) {
        System.err.println("There was a problem while reading the input data-sets: + e);
    }
}
```

// We may check if there are some missing attributes
somethingWrong = somethingWrong || train.hasMissingAttributes();

// Now we parse the parameters
nLabels = Integer.parseInt(parameters.getParameter(0));
String aux = parameters.getParameter(1); // Computation of the compatibility degree
Example

2. Execute the main process of the algorithm:
   • Abort the program if we have found some problem
   • Perform the algorithm's operations

   ```java
   public void execute() {
   if (somethingWrong) { // We do not execute the program
      System.err.println("An error was found, the data-set have missing values");
      System.err.println("Please remove those values before the execution");
      System.err.println("Aborting the program");
   }
   // We should not use the statement: System.exit(-1);
   else { // We do here the algorithm's operations
      nClasses = train.getnClasses();
      dataBase = new DataBase(train.getnInputs(), nLabels, train.getRanges(), train.getNames());
      ruleBase = new RuleBase(dataBase, inferenceType, combinationType, ruleWeight, train.getNames(), train.getClasses());
      System.out.println("Data Base:
" + dataBase.printString());
      ruleBase.Generation(train);
   }
   ```
Example

3. Write the output files:
   • The DB and the RB
   • Two output files with the classification for both validation and test files (doOutput)

```java
public void execute () {
    
    DataBase.writeFile(this.fileDB);
    ruleBase.writeFile(this.fileRB);
    // Finally we should fill the training and test output files
    double accTra = doOutput(this.val, this.outputTr);
    double accTst = doOutput(this.test, this.outputTst);
    System.out.println("Accuracy obtained in training: "+accTra);
    System.out.println("Accuracy obtained in test: "+accTst);
    System.out.println("Algorithm Finished");}
}

private double doOutput(myDataset dataset, String filename) {
    String output = new String(""); int hits = 0;
    output = dataset.copyHeader(); //we insert the header in the output file
    // We write the output for each example
    for (int i = 0; i < dataset.getnData(); i++) { //for classification:
        String classOut = this.classificationOutput(dataset.getExample(i));
        output += dataset.getOutputAsString(i) + " " + classOut + "nn";
        if (dataset.getOutputAsString(i).equalsIgnoreCase(classOut)) {
            hits++;
        }
    }
    Files.writeFile(filename, output);
    return (1.0*hits/dataset.size());
}

http://www.keel.es/software/Chi_source.zip
```
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
KEEL-dataset

It contains approximately 120 datasets from topics as diverse as credit risks, patients classification, sensor data of a mobile robot, ... Datasets with missing values and noise are included.

A recent development is the creation of the KEEL-dataset at http://www.keel.es/datasets.php
KEEL-dataset contains two main sections:

1. A detailed categorization of the considered data sets and a description of their characteristics. This contains a range of large and complex data sets for: classification (standard, low quality, imbalanced, Multi-Instance and with missing values), regression and unsupervised.
KEEL-dataset

2. A description of the papers which have used the partitions of data sets available in the KEEL-dataset repository. These descriptions include results tables, the algorithms used and additional material.
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works
- References about KEEL
Further works

- New block for Multi-Label Classification.

- A graphical tool to run in a distributed environment the experiments designed with the off-line module.

- We are developing a new set of evolutionary learning algorithms and a test tool.
Outline

- Introduction
- KEEL
- Why open source?
- KEEL-dataset
- Further works

References about KEEL
References about KEEL


Thank you!

Questions?