A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment

Ingo Mierswa, Ralf Klinkenberg, Simon Fischer, and Oliver Ritthoff

University of Dortmund, Germany

Department of Computer Science, Artificial Intelligence Unit

E-Mail: {fischer,klinkenberg,mierswa,ritthoff}@ls8.cs.uni-dortmund.de
WWW: http://yale.cs.uni-dortmund.de/

1 Requirements for Knowledge Discovery Platforms

Real-world knowledge discovery processes typically consist of complex data pre-processing, machine learning, evaluation, and visualization steps. Hence a data mining platform should allow complex nested operator chains or trees, provide transparent data handling, comfortable parameter handling and optimization, be flexible, extendible and easy-to-use. Depending on the task at hand, a user may want to interactively explore different knowledge discovery chains and continuously inspect intermediate results, or he may want to perform highly automated experiments off-line in batch mode. Therefore an ideal data mining platform should offer both, interactive and batch interfaces. In this paper, we propose YALE, Yet Another Learning Environment, which meets these requirements.

2 Modeling Knowledge Discovery Processes as Operator Trees

Knowledge discovery (KD) processes are often viewed as sequential operator chains. In many applications, flat linear operator chains are insufficient to model the KD process and hence operator chains need to be nestable. Consider for example a complex KD process containing a learning step, whose parameters are optimized using an inner cross-validation, and which as a whole is evaluated by an outer cross-validation. Nested operator chains are basically trees of operators. In YALE, the leafs in the operator tree of a KD process correspond to simple steps in the modeled process. Inner nodes of the tree correspond to more complex or abstract steps in the process, like e.g. cross-validation or feature selection. The root of the tree hence corresponds to the whole experiment. Operators define their expected inputs and delivered outputs as well as their obligatory and optional parameters, which enables YALE to automatically check the nesting of the operators, the types of the objects passed between the operators, and mandatory parameters.

YALE uses XML, a widely used language well suited for describing structured objects, to describe the operator trees modeling KD processes. XML has become a standard format for data exchange. Furthermore this format is easily readable by humans and machines. YALE can be started off-line, if the experiment configuration is provided as XML file, which can be edited with both arbitrary text or XML editors and with the YALE GUI. The YALE GUI provides a tree view corresponding to the XML document. The GUI allows the guided design of operator trees, interactive control and inspection of running experiments, and continuous monitoring of the experimental results.

Figure 1 shows a screenshot of the YALE GUI for a feature selection experiment using a genetic algorithm (GA) approach. The lower window at the left displays the experiment chain as operator tree and the upper window at the right depicts a result graph for the relative error of this approach in dependence on the generation number.

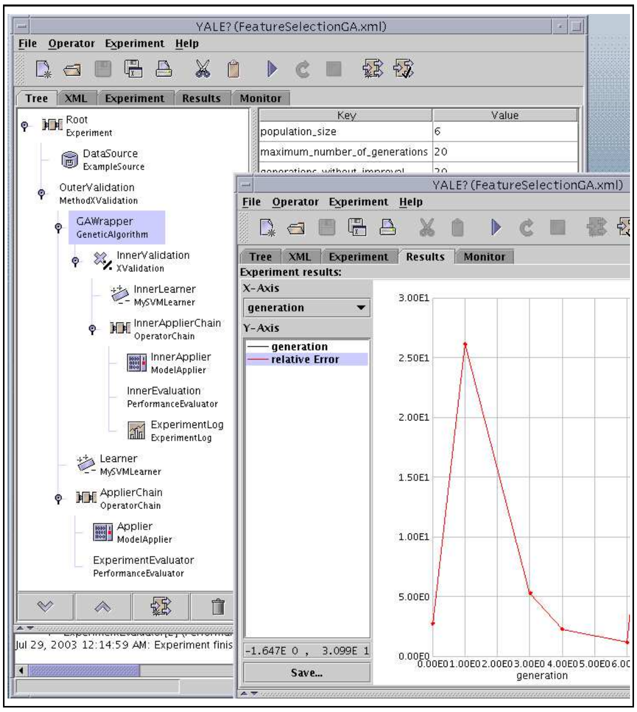

Figure 1: YALE screenshot: operator tree and result view.

3 Transparent Data Handling in YALE

Since operators define their expected inputs, delivered outputs, and parameters, YALE can automatically check their nesting, the types of the objects passed between them, and their mandatory parameters. YALE achieves a transparent data handling by supporting several types of data sources and hiding internal data transformations and partitionings from the user, i.e. one may simply exchange the data source or some other operator in an experiment and leave everything else unchanged. The supported data sources include pre-defined standard file formats like the ARFF format used by Weka [Witten and Frank, 2000] and CSV (comma-separated values), user-definable file formats, and SQL databases like Oracle, MySQL, and PostgreSQL. If operators generate data sets internally, these can easily be integrated into the YALE data flow.

3.1 Handling Input and Output Objects

The input objects of an operator may be consumed or passed on to following or enclosing operators. Especially if the input objects are not required by this operator, they are simply passed on. They do not even need to be interpretable by this operator. However, these input objects may be needed or used by later or outer operators. This increases the flexibility of YALE by easing the match of the interfaces of consecutive operators and allowing to pass objects from one operator through several other operators to their goal operator. Objects typically passed between operators are example sets, prediction models, evaluation vectors, etc. Operators may add information to input objects, e.g. labels to previously unlabeled examples, or new features in a feature generation operator, and deliver these extended objects. The specification of meta data is possible and allows for example the automatic selection of data pre-processing operators fitting to the data types at hand.

3.2 Efficient Data Management

No matter whether a data set is stored in memory, in a file, or in a database, YALE internally uses a special type of data table to represent it. In order not to unnecessarily copy the data set or subsets of it, YALE manages views on this table, so that only references to the relevant parts of the table need to be copied or passed between operators. By maintaining a stack of views, these views are nestable as is for example required for nested cross-validations For an example set, views on the rows of the table correspond to subsets of the example set, and views on the columns correspond to the selected features used to represent these examples.

4 Extending YALE

YALE supports the implementation of user-defined operators. The user simply needs to define the expected inputs, the delivered outputs, the mandatory and optional parameters, and the core functionality of the operator [Fischer *et al.*, 2003]. Everything else is done by YALE. The operator description in XML allows YALE to automatically create corresponding GUI elements. External programs can be integrated by implementing wrapper operators and can then be transparently used in any YALE experiment.

5 Example Applications and Download

YALE is used by researchers and practitioners in more than 20 countries and has already been applied in a number of domains like feature generation and selection [Klinkenberg *et al.*, 2002; Ritthoff and Klinkenberg, 2003; Ritthoff *et al.*, 2001; 2002], concept drift handling [Klinkenberg and Joachims, 2000; Klinkenberg and Rüping, 2003; Klinkenberg, 2003], and transduction [Daniel *et al.*, 2002; Klinkenberg, 2001]. Current application domains of YALE also include the pre-processing of and learning from time series [Mierswa, 2003] and processing and classification.

YALE is available as open-source software under the GNU Public License (GPL)[1]. The YALE tutorial [Fischer *et al.*, 2003], the GUI manual, and a long version of this paper [Mierswa *et al.*, 2003] provide further information about YALE, the underlying concepts, its usage, an operator reference, and how to define additional operators.

6 Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG), Collaborative Research Center on Computational Intelligence (SFB 531)[2] at University of Dortmund. We are grateful to Stefan Haustein for kxml, Stefan Rüping and Thorsten Joachims for their support vector machine implementations SVM^{light} and *mySVM*, respectively, the developers of Weka for their open source Java data mining library [Witten and Frank, 2000], and Katharina Morik and Timm Euler for their support. The SVM and Weka learners can be transparently used in YALE.

[1] `http://yale.cs.uni-dortmund.de/`

[2] `http://sfbci.uni-dortmund.de/`

References

[Daniel *et al.*, 2002] G. Daniel, J. Dienstuhl, S. Engell, S. Felske, K. Goser, R. Klinkenberg, K. Morik, O. Ritthoff, and H. Schmidt-Traub. Novel learning tasks, optimization, and their application. In H.-P. Schwefel, I. Wegener, and K. Weinert, editors, *Advances in Computational Intelligence – Theory and Practice*, pages 245–318. Springer, Berlin, 2002.

[Fischer *et al.*, 2003] S. Fischer, R. Klinkenberg, I. Mierswa, and O. Ritthoff. YALE: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02 (2nd edition), Collaborative Research Center on Computational Intelligence (SFB 531), University of Dortmund, Dortmund, Germany, August 2003.

[Klinkenberg and Joachims, 2000] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. 17th Int'l Conf. on Machine Learning (ICML-2000)*, pages 487–494, San Francisco, CA, 2000. Morgan Kaufmann.

[Klinkenberg and Rüping, 2003] R. Klinkenberg and S. Rüping. Concept drift and the importance of examples. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining – Theoretical Aspects and Applications*, pages 55–77. Physica-Verlag, Heidelberg, Germany, 2003.

[Klinkenberg *et al.*, 2002] R. Klinkenberg, O. Ritthoff, and K. Morik. Novel learning tasks from practical applications. In *LLA'02: Lehren – Lernen – Adaptivität, Proc. Workshop of the Special Interest Groups Machine Learning (FGML), Intelligent Tutoring Systems (ILLS), and Adaptivity and User Modeling in Interactive Systems (ABIS), German Computer Science Society (GI)*, pages 46–59, Univ. of Hannover, Germany, 2002.

[Klinkenberg, 2001] R. Klinkenberg. Using labeled and unlabeled data to learn drifting concepts. In *Workshop notes of IJCAI-01 Workshop on* Learning from Temporal and Spatial Data, pages 16–24, Menlo Park, CA, USA, 2001. AAAI Press.

[Klinkenberg, 2003] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 2003. To appear.

[Mierswa *et al.*, 2003] I. Mierswa, R. Klinkenberg, S. Fischer, and O. Ritthoff. A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment. Technical report, Collaborative Research Center on Computational Intelligence (SFB 531), Univ. of Dortmund, 2003.

[Mierswa, 2003] I. Mierswa. Beatles vs. Bach: Merkmalsextraktion im Phasenraum von Audiodaten. In *Proc. LLWA-2003/FGML-2003 – Annual meeting of the Special Interest Group on Machine Learning, Knowledge Discovery, and Data Mining (FGML) of the German Computer Science Society (GI)*, University of Karlsruhe, Germany, October 2003.

[Ritthoff and Klinkenberg, 2003] O. Ritthoff and R. Klinkenberg. Evolutionary feature space transformation using type-restricted generators. In *Proc. Genetic and Evolutionary Computation Conference (GECCO 2003)*, Berlin, 2003. Springer.

[Ritthoff *et al.*, 2001] O. Ritthoff, R. Klinkenberg, S. Fischer, I. Mierswa, and S. Felske. YALE: Yet Another Machine Learning Environment. In *LLWA'01/FGML-2001 – Proc. GI-Workshop-Woche* Lernen – Lehren – Wissen – Adaptivität, pages 84–92, Univ. of Dortmund, Germany, 2001.

[Ritthoff *et al.*, 2002] O. Ritthoff, R. Klinkenberg, S. Fischer, and I. Mierswa. A hybrid approach to feature selection and generation using an evolutionary algorithm. In *Proc. 2002 U.K. Workshop on Computational Intelligence (UKCI-02)*, pages 147–154, Univ. of Birmingham, UK, 2002.

[Witten and Frank, 2000] Ian H. Witten and Eibe Frank. *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufman, San Francisco, CA, USA, 2000. `http://www.cs.waikato.ac.nz/ml/weka/`.