

AI Tools: A Short Introduction to YALE

Arndt Faulhaber
Uhlandstr. 34, 66121 Saarbrücken
(arndt.faulhaber@diagnosdata.com)

April 29, 2007

Term paper elaborating the seminar talk
“YALE – Yet Another Learning Environment”.

Advised by: Michael Feld

Saarland University
Department of Mathematics and Computer Science
Institute for Computer Science

Seminar “AI Tools”
Winter term 2006/07.

Instructors:
Michael Kipp, Michael Feld, Dominik Heckmann and Alassane Ndiaye.

Abstract

In this term paper I introduce the machine learning framework called YALE (Yet Another Learning Environment). The paper motivates by introducing some aspects about the usefulness of machine learning in general and provides some examples of areas of its application. After that, I note some of the main aspects of machine learning frameworks followed by an in depth look at the inner workings of YALE, wherein I compare it to another open source machine learning framework called WEKA. I focus especially on the aspects that lead the creators of YALE to implement a new machine learning framework instead of using an existing one. These aspects are the way machine learning experiments are constructed and the way simple building blocks can be combined into more complex ones, some aspects about the way data is represented and handled internally, preprocessing and the overall open design of the framework. The second part of the paper is concerned with the seamless integration of YALE into one's own Java projects. After which I summarize the main differences of YALE and WEKA. A sideways look towards related work and a section where I conclude on which might be reasonable scenarios for the application of either YALE or WEKA, will complete this document.

Contents

1	Introduction	3
2	Machine Learning Frameworks	5
3	YALE – Yet Another Learning Environment	6
3.1	Experiment Structure	6
3.2	Operators	7
3.3	Preprocessing	7
3.4	Data in Memory Representation	7
3.5	Plugins	9
3.6	Additional Notes on YALE	9
4	Integrating YALE into your own Java Project	10
4.1	Using YALE from inside Java	10
5	A brief comparison between YALE and WEKA	12
6	Related Work	13
7	Summary and Conclusion	13

1 Introduction

Machine learning (ML), being a major topic in artificial intelligence (AI), has reached increasing importance in recent years. The exponential growth of processing capabilities¹ of modern computers and increasingly easy sharing of data using computer networks lead to the availability of enormous amounts of data – be it data gathered from scientific experiments or data from user interactions with computer systems or data simply contained in some kind of electronically available document, like PDF or others. Especially with the emergence of the world wide web (www) and its enormous growth lead to a vast amount of data being available at anyone’s fingertip. Now, how can anyone hope to dig through as much information in a reasonable amount of time? A possible solution is obviously to let computers do at least part of the work. That means we employ computers to single out information relevant to us and thus reduce the workload of manually inspecting every document or dataset.

Initially, in the early sixties, some of the first steps applying machine learning were aimed at pattern recognition as outlined in a great overview article by Marvin Minsky [3] – though, the early successes lead to an extremely high expectation towards AI in general. This expectation could not be met by research – namely reproducing human intelligence in a machine – and thus the so-called AI winter came to pass during the eighties (which hit the field of artificial neuronal networks especially hard).

Nevertheless, development of different kinds of machine learning algorithms went on and gathered momentum with the above mentioned emergence of computer networks. Nowadays ML is all around us. Go to Google for example and start a query. The search engine will return a number of links to web resources it deems relevant to your query in virtually no time (usually some fraction of a second). How can the response – a search taking into account a vast amount of web content – come so quickly? The answer is simple, though. Google makes heavy use of machine learning algorithms to analyze websites’ content and build up indices. A search on an index is a lot quicker than searching through each webpage individually. Currently Google maintains an index for over $8 * 10^9$ webpages² [7]. This particular application is called information retrieval. In general the task of automatically extracting knowledge from arbitrary documents is called *knowledge discovery and data mining* (KDD).

KDD is based on machine learning algorithms. These allow to single out criteria on how to classify a dataset, by training on some previously classified examples (in the case of supervised learning)³. ML is applied in many other areas that need classification or regression as well. Sorting out spam or the correct recognition of handwriting or speech are just some prominent examples.

Speaking of machine learning algorithms leads us to a huge diversity of approaches to classification and regression. Over the last decades many different algorithms for machine learning have been developed, ranging from easily interpretable ones featuring decision trees or rule based approaches⁴ to models that are rather hard to interpret, like support vector machines (SVM) or artificial neural networks (ANN)⁵ [1][4]. To simplify performance com-

¹In accordance to Moore’s law, stating that the computing power doubles about every two years.

²www.google.com, March 2007

³Note, that there exist also a number of algorithms for unsupervised learning, like clustering or SOMs (self organizing maps).

⁴Referred to as symbolic learning algorithms.

⁵Known as subsymbolic learning algorithms.

parison between all the different algorithms and allow their employment without the need to rewrite the whole experiment, different machine learning frameworks have been developed during the last decades. One of the oldest and most well-known freely available frameworks is WEKA, developed at the University of Waikato, New Zealand [6]. Work on WEKA started in 1993, funded by the New Zealand government. Another candidate, not yet as prominent as WEKA but quickly catching up, is YALE (Yet Another Learning Environment). Its development started at the University of Dortmund, Germany, in 2001. The intention of this project is to overcome some weaknesses identified in WEKA, like preprocessing and flexibility.

This work focuses on such frameworks which provide the necessary libraries for employing machine learning techniques to arbitrary learning tasks. The main focus herein is to analyze advantages and disadvantages of YALE and aims at discussing them in comparison to WEKA, since WEKA is a mature and widely used ML framework.

The first part of this paper will outline some of the intentions and features of ML frameworks in general followed by an introduction to YALE. In the next section of this article an example of how to easily integrate YALE into one's own Java-Project without losing any flexibility YALE is offering will be presented. The third section will state a brief comparison between YALE and WEKA. Finally I will describe some of the related work and summarize this article with some conclusions that can be drawn from the comparison between YALE and WEKA.

2 Machine Learning Frameworks

ML frameworks, as mentioned earlier, supply a variety of different machine learning algorithms as well as the possibility to combine these into complex experiment setups. For example one can make the framework choose a specific learning algorithm based on its performance (that can be assessed using k-fold cross validation and a variety of performance metrics, like κ or precision) or combine multiple algorithms into a meta-learner with automatic weighting of each algorithm. Often these frameworks provide a graphical user interface to create, modify and run such experiments. Additionally to the handling of experiments, it is often possible to inspect or visualize the processed data at different steps of the experiment, for example as a way to debug or verify the proper working of the experiment. Some frameworks even provide their own programming language as is the case with R⁶.

Today, a variety of frameworks are available – some commercial, others free and open source.

The focus of such frameworks varies a lot, from pure statistical viewpoints over more concrete application areas, e.g. data mining and knowledge discovery, to very specialized systems that concentrate on some specific area of application like voice recognition. YALE and WEKA as the main focal points of this text try to provide an open framework for a wide range of applications by incorporating a multitude of different learning algorithms, assessment methods and associated tools.

Visualization is one further aspect of importance. Visualizing certain aspects of a problem may greatly help to solve it – by providing the possibility to view the same problems from different angles or different related problems from the same angle, thus making them comparable. The scope and granularity of visualization facilities within machine learning frameworks is extremely diverse, as with most other aspects already mentioned.

A reasonable decision on which machine learning framework is best used for a specific task, often depends on the task itself. Some frameworks provide APIs for different languages (like Java or C), others are embedded into some Program and can only import data and some even provide services via network servers, that will handle requests by some client program and return the processed data.

The following section takes a deeper look at YALE, one of the freely available ML frameworks written in Java, and tries to illuminate some of its inner workings with regard to usability, flexibility and performance.

⁶<http://www.r-project.org/>, last visited March 2007

3 YALE – Yet Another Learning Environment

YALE is a framework for knowledge discovery and data mining (KDD) and machine learning. Work on the YALE project started in 2001 at the University of Dortmund, at the chair of Artificial Intelligence. The first version was released in 2002 and the project has been moved to SourceForge⁷ in 2004. It is a library written in Java under the GPL. The current version of YALE is 3.4 and to this the text refers.

The authors called their project “Yet Another...” to indicate, that they are aware of existing machine learning frameworks, but have various reasons for why they chose to implement a new ML framework, namely the lack of a suiting, freely available library for ML/KDD that offers:

- Integrated easy preprocessing of data
- Advances customizability, e.g. automatic selection of the best performing classifier
- A simple way to vary the datasets used in the experiment (e.g. for cross-validation)
- Flexible attribute creation and selection

Looking at recent developments in WEKA, some of these points have already been addressed in newer versions. The current version of WEKA is 3.4.10 and will be used as reference version throughout this text.

Some other considerations have been pointing towards deeper aspects of WEKA, like in-memory copying of data or the way data is represented.

3.1 Experiment Structure

Both WEKA and YALE provide a GUI for easy and rapid development of experiments. An Experiment in this context means the whole process of loading data into the ML framework, defining and selecting attributes, creating and applying classifiers, performing performance analysis and even visualizing the results.

While WEKA provides a more linear representation to building an experiment where the user can enable certain steps of an experiment, like cross-validation, YALE provides a different view in experiments. Experiments are represented in a hierarchical order, i.e. a tree-like structure called operator tree [2]. Figure 1 shows how such an operator tree looks like in the YALE-GUI.

Any subtask in YALE can consist of an arbitrary number of subtasks which again can be composed of more subtasks and so forth. This approach provides the ability to create new metalearning schemes by combining “simple” components into more complex ones. For example, you can create a new operator that selects relevant attributes based on a genetic algorithm. Such subtasks can be reused in a new experiment such that previous work can be easily integrated into a new experiment.

⁷<http://www.sf.net/> an open source development platform.

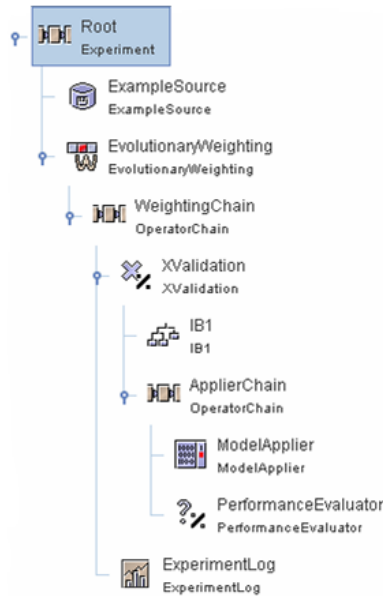


Figure 1: Example: small operator tree in YALE

3.2 Operators

Operators are the basic building blocks used in YALE [2]. YALE knows two different kinds of operators, basic operators that represent specific classes (e.g. a classifier or performance counter) and operator chains which are containers for multiple basic operators. Operators are processed in the way of a depth first search through the tree with the exception of loops which will be traversed multiple times. Since every operator has a defined type of input and output it is quite easy to debug the flow of data from one operator to the next and, when using the GUI, the correctness will be verified upon a single mouse click. Additionally, YALE allows the user to set breakpoints before, within or after each operator. Thus, one can inspect the data transformation at each single step in the experiment.

3.3 Preprocessing

To address the issue of preprocessing data, YALE integrates various operators for loading data from different sources (e.g. various file formats, databases or even websites using a plugin). As shown in figure 2, it is possible to specify regular expressions to parse specific parts of the input, specify column delimiters, comment tokens and so forth.

Moreover, it is quite easy to write one's own preprocessing class and integrate it either directly into YALE or as a plugin.

3.4 Data in Memory Representation

One of the main points criticized in WEKA is that a lot of in memory copying takes place during machine learning. This inspired the YALE people to implement a generic way to represent data in memory, called *multi-layered data view* as described in [2][5]. Views can be

Key	Value
configure_operator	Start configuration wizard...
attributes	C:\Programme\YALE\yale-3.4\ls Edit ...
sample_ratio	1.0
sample_size	-1
datamanagement	double_array ▼
column_separators	,\s* ;\s*\ s+
comment_chars	#
decimal_point_character	.
use_quotes	<input type="checkbox"/>
permutate	<input type="checkbox"/>
local_random_seed	-1

Figure 2: Example: Preprocessing in YALE

seen as an analogy to database system views in that only part of the data is visible. Figure 3 shows how views are organized.

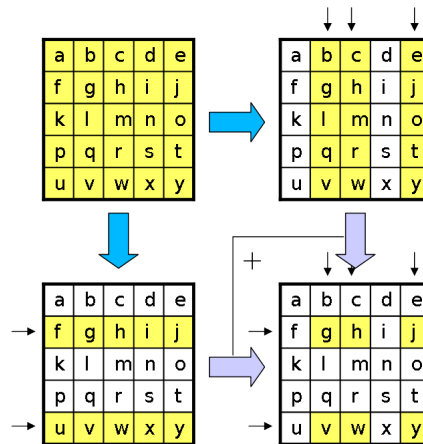


Figure 3: Example: Multi-layered data view in YALE

YALE keeps a stack of views (that act as filters) on a data table (the complete table of instances and their attributes) and only passes references to relevant parts of the table. By offering different ways to filter data without copying specific parts an increase in performance and a smaller memory footprint should be achieved.

Furthermore, YALE provides a uniform interface to data independent of its location. That means references to data stored in a database look exactly the same as references to data in a file, thus simplifying change of data sources and permitting heterogeneous data sources without the need to reimplement or adjust the rest of the experiment. This is useful especially for cross-validation since many different views on the dataset are needed to form the different training-/testset partitions.

3.5 Plugins

Flexibility through extensibility as one of the goals of YALE is introduced in two ways:

- A plugin mechanism
- By means of writing just a new operator that implements YALE's operator interface

A number of plugins are available for YALE. These include a word vector tool that automatically creates a table of words and their frequency of occurrence in a specified document or webpage. Others allow for time series analysis or clustering⁸. How to create a plugin is documented in the YALE Tutorial [10], that can also be downloaded from the YALE website [9].

3.6 Additional Notes on YALE

Some additional things may be interesting to get a good picture of YALE. First of all, it needs to be mentioned that YALE integrates wrappers to WEKA's classifiers and thus offers the complete variety of WEKA's classifiers. Secondly, at the time this document was written, the official version of YALE did not support online learning (i.e. incremental training of an existing model using new samples). Though this has been addressed in the forum and was promised to be incorporated into YALE as soon as possible⁹. A patch has been integrated into the current CVS version which now includes a ModelUpdater operator addressing this issue.

⁸Currently 6 plugins are officially available at <http://rapid-i.com/>, March 2007

⁹The feature request has been submitted on January 30, 2007, as forum post http://sourceforge.net/forum/forum.php?thread_id=1660962&forum_id=390413.

4 Integrating YALE into your own Java Project

In the following, I will show a possible way to integrate YALE into one's own Java project without losing any flexibility provided by the YALE GUI as rapid development platform for data analysis tasks. This process includes two parts:

- Building experiments using the YALE GUI
- Including YALE into the project to load and execute experiments

Since the structure of experiments has been previously dealt with and the YALE GUI Manual [11] offers a detailed description on how to create experiments, I herein will only describe how to use YALE from inside a Java project and use two simple predefined experiments.

4.1 Using YALE from inside Java

The first step in integrating YALE is to provide two experiments, one for training a model and one that will use the trained model to do the actual classification. Figures 4 and 5 show two sample experiments that will perform each task respectively.

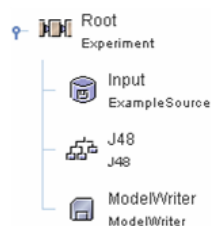


Figure 4: Example: a simple YALE experiment for model creation

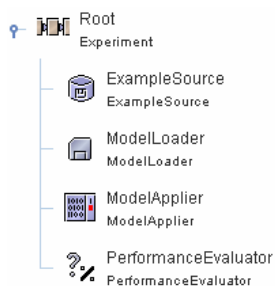


Figure 5: Example: a simple YALE experiment for classification

The next step is to add YALE as a library (or include the source code) to the project. And then the actual integration in the code can start.

Before YALE can work properly it has to be initialized:

```
// Initialization of YALE
System.setProperty("yale.home", "C:\\Programme\\YALE\\yale-3.4");
Yale.init();
```

After YALE is initialized, an experiment can be loaded:

```
//*****
// load Experiment - The Model creation
//*****
File expFile = new File("yale-test/MyExperiment.xml");
Experiment e = new Experiment(expFile);
```

and executed:

```
// start Experiment
IOContainer res = e.run();
```

Thereafter, depending on the experiment output, the different parts of the output can be accessed in the IOContainer. For example, the model can be retrieved and put into a new experiment or performance evaluation data can be extracted and displayed.

The advantage of this approach lies in the ability to use the GUI for rapid development of learning experiments, with all its advantages, without the need to change a single line of code in the Java program. Since the new model will be loaded and applied during the next run of the application. It is also possible to load a new model without restarting the application, as demonstrated in our¹⁰ spam filter project developed as a practical assignment during the AI Tools course.

¹⁰Adrian Schröter's and mine

5 A brief comparison between YALE and WEKA

After describing some of the internal workings of YALE, I want to give a short summary on the comparison between YALE and WEKA.

As the main focus of the initiators of YALE has been flexibility, it is quite reasonable to expect most changes in this area. Namely flexibility in creating experiments, reusing previous work and preprocessing data to work on. These criteria have obviously been addressed by:

- The openness of arranging experiments using operator trees
- A single coherent interface to the experiment data employing multi-level data views and
- A multitude of preprocessing filters integrated into the framework.

The process of integrating the framework into Java projects seems to be easy enough for both YALE and WEKA and is outlined for both at length in the respective documentation or community resources¹¹, though reusing experiments created in the WEKA Experimenter should be possible but the available documentation on this is somewhat scarce.

Comparing the availability of classifiers between YALE and WEKA, YALE is somewhat more extensible in that it offers the possibility to use wrappers to attach even non-Java classifiers into the framework, otherwise most classifiers provided with YALE are actually those from WEKA. But as YALE gains momentum, it is possible that this will quickly change as other people start to integrate additional classifiers into YALE.

Regarding performance, WEKA in spite of in memory copying of data still seems to run fast enough and since it provides the ability to run in a distributed environment¹², it still seems an adequate choice for huge experiments.

¹¹In finding the relevant information for WEKA a bit more searching is involved, but there are quite a few websites featuring howtos and tutorials on how to integrate WEKA into one's own Java programs

¹²A number of web resources describe how to do this (e.g. <http://smi.ucd.ie/rinat/weka/>, last accessed March 2007)

6 Related Work

Quite a number of frameworks for machine learning and data mining are available by now, though most of these are commercial, like SPSS Clementine¹³. Many Commercial frameworks are quite limited in scale as they provide only a limited number of classifiers or meta operators like cross validation. In the open source area, WEKA, YALE and Tanagra¹⁴ are probably the most complete and extensible frameworks available, but a number of standalone classifiers like mySVM¹⁵ or programming libraries like ADaM¹⁶ exist. Another approach at data analysis is R¹⁷, which is heavily used for statistical analysis of data and also features a number of machine learning algorithms. Being very efficient with handling data, it completely lacks a graphical user interface for building projects and uses a rather unintuitive syntax one has to get used to. Furthermore, the integration into one's own projects is probably quite hard since it cannot be used as a library.

7 Summary and Conclusion

We have seen that YALE addresses the problems that initially lead the people behind YALE to create yet another learning environment, namely a flexible framework providing many helpful components for integrated preprocessing, advanced customizability by implementing an operator hierarchy in which new meta operators can be easily created and reused, a simple way to vary the datasets used in the experiment providing multi-layered data views and finally flexible attribute creation and selection again via use of the operator hierarchy.

Summarizing I can only state that, as far as I can judge, both frameworks, YALE and WEKA, deliver a rich platform for machine learning tasks, providing a multitude of bleeding edge machine learning components and a good graphical environment for rapid development of learning projects.

After some hands-on experience with both frameworks, it is hard to vote in favour for either one. My personal opinion is that the programming interface and data structures seem a bit tidier in YALE and the concept more coherent. But WEKA as the far older framework – none the less actively developed – seems better documented and perhaps more robust. Thus I conclude to choose a framework based on the requirements of the experiment or solution which has need of machine learning. YALE seems to be a good choice for tasks that bear ongoing change of structure in the data, where heavy preprocessing is involved and the models frequently have to be adjusted. Whereas WEKA might be the better choice for projects, that rarely change in nature or that convey heavy processing need imposing the use of a distributed computing environment (like cluster or grid computing).

¹³<http://www.spss.com/clementine/>, last accessed March 2007

¹⁴<http://chirouble.univ-lyon2.fr/ricco/tanagra/>, last accessed March 2007

¹⁵<http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html>, last accessed March 2007

¹⁶<http://datamining.itsc.uah.edu/adam/>, last accessed March 2007

¹⁷<http://www.r-project.org/>, last visited March 2007

References

- [1] T. Hastie, R. Tibshirani and J. Friedman: “The Elements of Statistical Learning”, Springer, 2001.
- [2] I. Mierswa et al: “YALE: Rapid Prototyping for Complex Data Mining Tasks”, 2006.
- [3] M. L. Minsky: “Steps Toward Artificial Intelligence” Proc. IRE, Vol. 49, pp. 8-30, 1961.
- [4] T. Mitchell: “Machine Learning”, McGraw Hill, 1997.
- [5] O. Ritthoff et al: “YALE: Yet Another Learning Environment”, 2001.
- [6] I. H. Witten and E. Frank: “Data Mining: Practical machine learning tools and techniques”, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

Web-references:

- [7] <http://www.google.com/>, homepage of the popular web search engine. Last visited March 2007.
- [8] <http://www.cs.waikato.ac.nz/ml/weka/>, homepage of the WEKA project. Last visited March 2007.
- [9] <http://rapid-i.com/>, homepage of the YALE project. Last visited March 2007.
- [10] http://rapid-i.com/component/option,com_weblinks/task,view/catid,24/id,27/lang,en/, YALE Tutorial. Last visited March 2007.
- [11] http://rapid-i.com/component/option,com_weblinks/task,view/catid,24/id,26/lang,en/, YALE GUI Manual. Last visited March 2007.