Information Miner – a Data Analysis Platform

Frank Rügheimer
Inst. for Knowledge and
Language Engineering
University of Magdeburg
ruegheim@iws.cs.uni-magdeburg.de

Rudolf Kruse
Inst. for Knowledge and
Language Engineering
University of Magdeburg
kruse@iws.cs.uni-magdeburg.de

Abstract

Knowledge discovery and information mining have long been recognized as important tools for prediction and decision making. But the process – from data preparation and selection of appropriate methods to a properly configured, well-tested analysis setup – can be time-consuming. In recent years easy to use software that supports experts in model construction has become available. Advocating a vertical system, this paper will introduce a toolkit for the construction of complex application-specific information mining solutions from components. Moreover, it outlines how extensions and suggested concepts may contribute to further improving support for the information mining process.

Keywords: data mining, information mining, application software

1 Introduction

The field of data mining and knowledge discovery in databases has become a highly successful area with respect to applications in both research and business. By now its importance has been broadly recognized for its potential to discover hidden relationships, e.g. in customer or process data, helping companies to improve their production or marketing strategies. Conversely, by pointing out relevant problems and directions for further development, numerous applications have influenced the field itself, so tools for supporting the information mining process have evolved. A focus of current development, information mining deals with extracting knowledge from various types of sources including multimedia data. The work described in this paper aims at providing adequate tools for efficient information mining in specialized applications.

2 Previous Approaches

In an initial phase data mining software was focused on single methods that were usually made available without a graphical user interface or support for the other phases of the data-mining process. The C++ library MLC++ [8] for example provides a collection of algorithms. But adaptations for specific applications and integration into a comprehensive solution are left to the user. As the importance of supporting the users became more thoroughly recognized, systems that combined single data analysis methods with a graphical user interface and integrated preprocessing tools appeared. Answertree (SPSS) and Business-Miner (Business Objects) are representatives of this class. Nowadays software packages usually combine several methods in one environment e.g. Clementine (Integral Solutions Ltd.), DeltaMiner (Bissantz & Co. GmbH).

Finally the need for specialized solutions has lead to systems which permit later extensions (Enterprise Miner (SAS)). Unfortunately the implementation of these extensions is often restricted by system architecture and development is bound to a fixed programming language. An alternative strategy defines transfer interfaces only, allowing for more flexibility and the integration of existing implementations using wrapper classes. This plug-in approach is used for instance in DataEngine (Management Intelligenter Technolo-

gien GmbH). The higher flexibility comes at the cost of additional data transfer operations though.

Since real world information mining problems are manyfold, the advantages of access to an extensive library of reusable methods and data exploration tools are easy to see. For some methods efficiency requirements may suggest an integrated implementation. Nevertheless, reuse of algorithms that have already been adapted to a distinct problem is often rewarding. Thus direct support for this method of expansion seems desirable.

3 Information Miner Platform

Information Miner provides a software construction toolkit that allows experts to easily develop, modify, and configure domain specific streams for data processing and analysis. We intended to combine the user-friendly features of general Information Mining Systems with the flexibility of module based approaches. The streams can be saved for later application by users who need not have a background in information mining. End-users can then easily apply previously generated models to up to date input data or even create models on their own.

In its basic version the software comes with modules for calculating statistics and a selection of standard methods e.g. decision trees [9], regression trees [4], Bayesian classifiers. Further learning methods include neural networks (currently multilayer perceptrons and radial basis function networks), association rules [1], and a selection of clustering approaches. Apart from learning algorithms the repository contains operations for data preprocessing, visualization, and model evaluation. While the user interface is written in Java, the implementation of algorithms is not restricted to this programming language. Command line parameters for calling external pieces of software are passed from the graphical user interface so existing programs can be integrated into the system. The platform itself can be configured by providing repositories that contain different combinations of analysis methods and support tools. The current version of Information Miner and its basic method repository have successfully been tested under Linux and MS-Windows operating systems.

3.1 Pipes and Filter Concept

The process of obtaining information from data comprises a sequence of tasks. Data have to be collected, combined, and preprocessed before being used for generating models. Algorithms and models must in turn be configured with appropriate parameters, tested, and evaluated before finally being applied to new data. In Information Miner processing streams are represented as directed acyclic graphs. Operations like e.g. reading or visualizing data, generating or applying models, or computing evaluation measures are symbolized by nodes. Users select appropriate operations from a component library and place the corresponding nodes onto a workspace. Following that, the nodes are connected with edges, which represent data or even whole models being passed between them (Fig. 1). Planned extensions include mechanisms to pass parameters as well.

Before processing a stream the graphs is analyzed to ensure that the tasks associated with each node are executed in correct order. Intermediate results are stored for later reuse. Thus, when carrying out modifications, only those processing steps that depend on reconfigured nodes or are affected by structural modifications to the stream have to be updated.

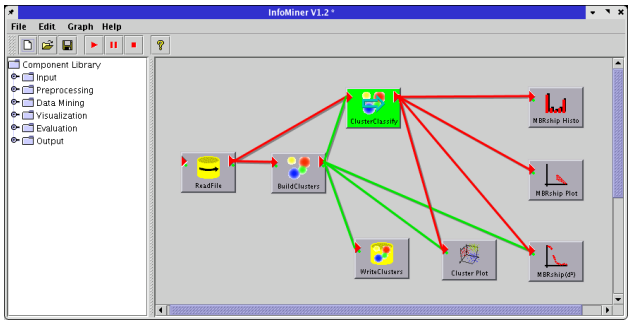

Figure 1: An example processing stream

The graphical representation has a number of advantages:

- efficient composition of complex application-specific data mining solutions,

- increased comprehensibility,
- automatic transfer of data and models via connecting arcs,
- reusability of partial solutions by adapting existing streams,
- direct access to method parameters via configuration dialogs.

Moreover, when combined with a suitable method repository, the graphical representation contributes to further enhancements e.g.

- immediate model evaluation using specialized nodes,
- fast access to implementations of standard methods for comparison,
- interchangeability of data sources.

3.2 Component Configuration

Once placed on the workspace, nodes have to be connected to form processing streams. Model configuration and data transfer within Information Miner are based on a flexible interface. Each processing node possesses a connector for expected inputs (models, parameters or data) of a specified basic types or Java Classes. Optional outgoing connectors provide data or models for subsequent nodes. When users add a new connection the system checks the modified graph and reject the edge if it would generate a cycle. Otherwise it attempts to match the outputs of the preceding node to the unassigned inputs of the subsequent one. Parameters are initialized with standard values, so working configurations are usually provided even without further user interaction. For adjustments and to provide inputs that cannot be determined automatically a generic configuration dialog is available for each node (Fig. 2). Table 1 gives explanations for the particular fields.

3.3 Integration of External Software

When implementations for specialized information mining algorithms are already available, it

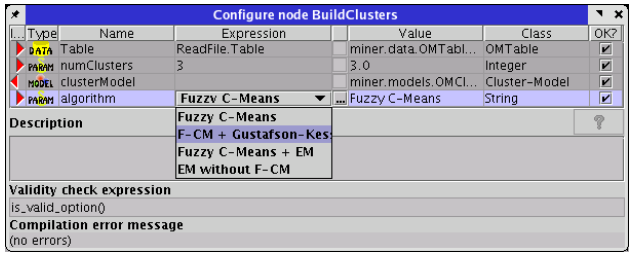

Figure 2: Dialog for Configuring Tasks and External Applications

Table 1: Parameters Configuration

I/O	Direction of information flow (input or output)
Type	Either 'Data', 'Model' or 'Parameter'
Name	Unique name for use in expressions
Expression (Inputs only)	Inputs are specified either directly or computed from expressions (generation of entries can be supported with lists of possible alternatives or additional dialogs, e.g. for browsing file systems). The result type is determined by the 'Class' field
Value	Result of evaluation
Class	Expected basic type or Java class
OK	Indicates if the selected value is consistent with restrictions for the particular type of node

is frequently profitable to integrate them into the information mining system such that existing partial solutions can be reused and combined with features from the toolkit. A simple, but effective approach consists in providing wrapper classes for external programs. Wrapper classes connect to the system's interface to obtain data and parameters via standard node configuration dialogs. The inputs are then passed on to the external applications. This strategy fosters reuse of software and provides an efficient way for extending the method library.

When the cost of data transfer is not considerable compared to other computations this method sometimes constitutes the preferred option. As the implementation of the algorithms is not bound to Information Miners internal data structures,

the data can be reorganized with respect to efficient access and processing by the method's specific algorithms.

4 Example Scenario

To demonstrate these concepts, one can consider a simple clustering task. For this example the iris data set, which contains data on flower geometry for specimen of three species of the iris family is used. The data was split in two separate files with 120 and 30 examples respectively, the smaller file being reserved for evaluation purposes.

To read the data file specific import nodes can be selected from the repository bar on the right and dragged onto the desktop. Using a filesystem browser each of the nodes can be connected to the desired input file. The designation of the imported tables is generated from the file names. In order to take a first look on the data a 'ScatterPlot' node is selected from the available visualization methods and connected to the output of the file import node (Fig. 3 left). Starting stream execution a visualization window opens (Fig. 3 right).

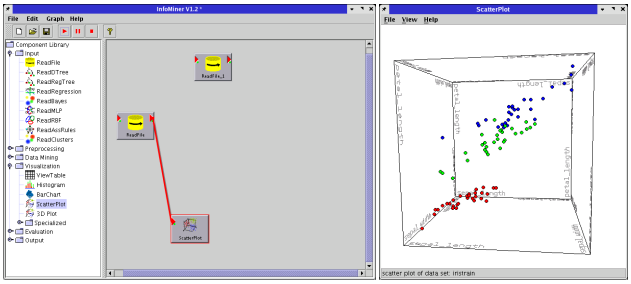

Figure 3: File Import and Visualization

In the next step we add the 'BuildClusters' node to generate cluster descriptions from the data. By default the last column of a table is expected to contain class lables and it will be ignored by the clustering module. Using an additional 'FieldOps' node one can overide these setting and assign different roles to each variable or even delete columns. In this example we just accept the defaults. For a first run we also use the default clustering settings, namely standard Fuzzy c-means algorithm (FCM) with three clusters. In order to assign the cluster membership to the training data an additional processing node is used and for assessment of the results we generate plots of the membership degrees (Fig. 4). The histogram node is added to show the distribution of the tree iris species over the classification achieved by grouping the examples according to the highest cluster membership.

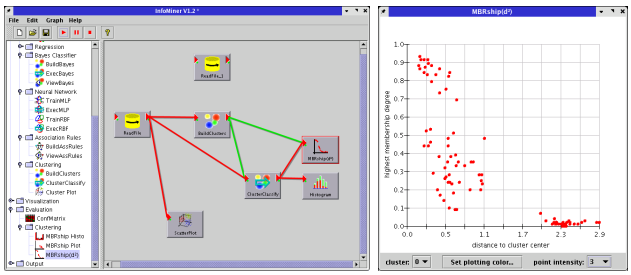

Figure 4: A Configuration for Clustering

It is soon revealed that two of the clusters are not clearly separated. Since elliptical clusters might be a better choice here, we reconfigure the clustering module for the Gustafson-Kessel algorithm [6] (see Fig. 2). Once model generation is configured, the result should be evaluated on the test data. The required nodes can be connected directly to the 'BuildClusters' node and it is only required to execute tasks for the new branch of the network graph. To allow later reuse the cluster description is also written to a file by sending it to a specialized output node. The resulting final graph is shown in Fig. 5.

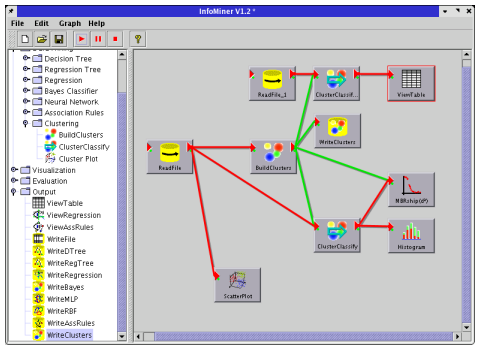

Figure 5: Final Example Stream

5 Consideration of Information Mining Challenges

Compared to traditional data mining, the input being processed in information mining can origi-

nate from a larger variety of sources and is usually not uniformly structured to begin with. A related problem is caused by large numbers of variables leading to large domains. While the problem has been known in traditional data mining, requirements of recent applications, fusion of inhomogeneous data and the frequent necessity to propositionalize multirelational databases, can aggravate it.

Finally, with respect to applications, it should be considered that users often demand intelligible models. Intelligible models allow them to check for consistancy with previous experience and thus gain confidence in the results. It also helps to recognize situations, in which an initially selected method is insufficient for the desired application. This sections discusses how the development of Information Miner's method library and transfer mechanism can accommodate these requirements.

5.1 Supporting Heterogeneous Data

While implementations of data mining algorithms are often designed to work with a homogeneous input, data encountered in many real world applications do not usually meet this requirement. Expanding the focus to information mining, heterogeneity may even extend to media. In vertical systems like Information Miner the problem of different data representations and sources is addressed by providing a collection of predefined import modules. Depending on representation import modules for text, XML, or databases can be used. When importing from multirelational databases propositionalization may be necessary before learning methods can be applied. Finally working with images, audio or video data necessitates feature extraction.

Preprocessing is complemented with learning methods that directly deal with inhomogeneous data e.g. semi-supervised learning of classificators [7]. These methods use both examples and the structure of yet unclassified input to arrive at better models. Such methods are preferred, when obtaining pre-classified training examples is costly. Finally modern data fusion techniques are instrumental in using the available information to full capacity. For measurements collected from a mixed ensemble of sensors it may be appropriate to use separate methods for processing the input from the individual sources and combine the results. Methods like Boosting [10] or Stacking are examples of multi-model approaches that can be used to combine different algorithms.

5.2 Fuzzy Rules for Large Feature Spaces

Many relevant applications of information mining e.g. analysis of gene expression patterns inherently deal with high dimensional data. Similarly multimedia data analysis and multirelational database mining usually involve dealing with high dimensional feature spaces.

In addition to that, data fusion and the propositionalization of multirelational databases, which are essential to accessing data previously unavailable for many analysis methods, usually produce high dimensional data sets. Although the number of input variables can sometimes be reduced with preprocessing, it is necessary to include methods that are robust to high dimensional input data. But while large feature spaces have to be searched, interesting relationships in such data often involve smaller subsets of variables and can comprehensibly be represented by fuzzy rules.

Fuzzy rules are easily understood by the users thus fulfilling their immediate information needs. The fuzzy rule induction algorithm given in [2] is specifically suited to dealing with large feature spaces and heterogeneous data. An advanced version [5] constructs a hierarchy of rule sets with different levels of complexity. For instance this approach allows users to assess basic relations with relatively coarse fuzzy rules while using a higher level of detail for prediction tasks. The algorithm has already been integrated into an experimental method repository for a previous version and will be adapted for the final interface.

5.3 Planned Extensions

Allthough many methods are already available for Information Miner, the software construction toolkit is continuously supplemented with implementations of new methods. A module for semi-supervised clustering is already being developed. We also plan to extend the transfer mechanism

for models and data to parameters. Besides simplifying method configuration, this also constitutes an intermediate step towards the grouping of complex or combined operations into supernodes, which could export the same interface as conventional processing nodes. The mechanism for passing parameters would allow to better combine modules and exchange relevant parameters with the external interface of the supernode. This concept would contribute to even better usability of the system. Capsulating preprocessing steps for instance, can hide complexity from the end-user. Furthermore supernodes are suited for elegant implementations of Bagging [3]. Finally the integration of external software could be supported by a wizard for creating wrapper classes.

6 Summary

Information Miner is a system designed to support a vertical approach to application specific information mining problems. By providing a software construction toolkit and a method repository, the information mining process can be supported. In combination with interfaces for the integration of existing software modules it contributes to considerably reducing the effort required for developing application specific solutions. The pipes & filter concept for interconnection is applied not only to data but also to models so solutions become more flexible. Addressing the problem of inhomogeneous data, access to preprocessing operators is supplemented with specialized algorithms that directly deal with inhomogeneous data. At the time this paper is written implementations of such algorithms are being prepared for integration into the method repository. Similarly the occurrence of high dimensional input data in many practical applications is managed by providing robust (fuzzy) rule based analysis methods.

References

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[2] M. R. Berthold. Induction of mixed fuzzy rules. *International Journal of Fuzzy Systems*, 3(2):382–389, 2001. (invited paper).

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression tree*. Wadsworth International Group, Belmont, CA, 1984.

[5] T. R. Gabriel and M. R. Berthold. Constructing hierarchical rule systems. In M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, editors, *Proc. 5th International Symposium on Intelligent Data Analysis (IDA 2003)*, Lecture Notes in Computer Science (LNCS), pages 76–87. Springer Verlag, 2003.

[6] D. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proc. of the IEEE Conference on Decision and Control*, pages 761–766, 1979.

[7] A. Klose and R. Kruse. Information mining with semi-supervised learning. In *Advances in Soft Computing: Soft Methodology and Random Information Systems*. Springer-Verlag, Berlin, Heidelberg, 2004.

[8] R. Kohavi. *Wrappers for performance enhancement and oblivious decicion graphs*. PhD thesis, Stanford University, 1995.

[9] J. R. Quinlan. Induction of decision trees. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[10] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proc. 13th National Conf. on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conf.*, pages 725–730. AAAI Press / MIT Press, Menlo Park, August 4–8, 1996.