Paper, presented at: Open Source in Data Mining Workshop, Bangkok, Thailand, 27 April 2009

The Konstanz Information Miner 2.0

Thorsten Meinl, Nicolas Cebron, Thomas R. Gabriel, Fabian Dill, Tobias Kötter, Peter Ohl, Kilian Thiel, Bernd Wiswedel, and Michael R. Berthold

University of Konstanz,
Nycomed Chair for Bioinformatics and Information Mining
Box 712, 78457 Konstanz, Germany
contact@knime.org

Abstract. In December 2008, version 2.0 of the data analysis platform KNIME was released. It includes several new features, which we will describe in this paper. We also provide a short introduction to KNIME for new users.

1 KNIME Overview

The Konstanz Information Miner (KNIME) is being developed by the Nycomed Chair for Bioinformatics and Information Mining at the University of Konstanz since 2004. KNIME is open source and available under a dual licensing scheme. Usage of KNIME is free for everyone providing no profit is made from distributing KNIME or selling extensions. Therefore for researchers, be it at universities or in companies, it is a free yet still powerful tool to perform all kinds of data analysis.

A previous paper[1] has already described KNIME's architecture and internals. In this paper we will therefore concentrate on the new features in the latest KNIME 2.0 release. For readers not yet familiar with KNIME, we shall provide a short overview of KNIME's key concepts.

With KNIME, the user can model workflows, which consist of *nodes* that process data that is transported via connections between the nodes. A flow usually starts with a node that reads in data from some data source, which are usually text files, but data bases can also be queried by special nodes. Imported data is stored in an internal table-based format consisting of columns with a certain data type (integer, string, image, molecule, etc.) and an arbitrary number of rows conforming to the column specifications. These data tables are sent along the connections to other nodes that first pre-process the data, e.g. handle missing values, filter columns or rows, partition the table into training and test data, etc, and then in most cases predictive models with machine learning algorithms like decision trees, naive Bayes classifiers or support vector machines are built. For inspecting the results of an analysis workflow several view nodes are available, which display the data or the trained models in various ways. Figure 1 shows a small workflow with some nodes.

In contrast to many other workflow or pipelining tools, KNIME nodes first process the entire input table before the results are forwarded to successor nodes.

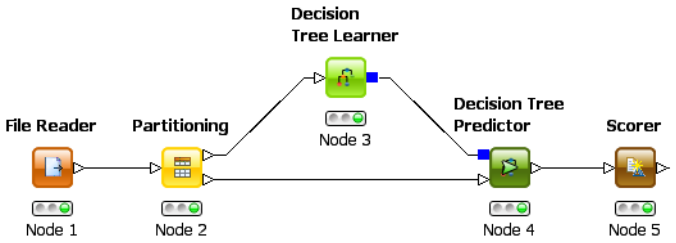

Fig. 1. A small KNIME workflow that builds and evaluates a decision tree.

The advantages are that each node stores its results permanently and thus workflow execution can easily be stopped at any node and resumed later on. Intermediate results can be inspected at any time and new nodes can be inserted and may use already created data without preceding nodes having to be re-executed. The data tables are stored together with the workflow structure and the nodes' settings. The small disadvantage of this concept is that preliminary results are not available as soon as possible as if real pipeling were used (i.e. sending along and processing single rows as soon as they are created).

One of KNIME's key features is *hiliting*. In its simple form, it allows the user to select and highlight several rows in a data table and the same rows are also hightlighted in all other views that show the same data table (or at least the hightlited rows). This type of hiliting is simply accomplished by using the 1:1 correspondence between the tables' unique row keys. However, there are several nodes that completely change the input table's structure and yet there is still some relation between input and output rows. A nice example is the *MoSS* node that searches for frequent fragments in a set of molecules. The node's input are the molecules, the output the discovered frequent fragments. Each of the fragments occurs in several molecules. By hiliting some fragments in the output table all molecules in which these fragments are contained are hilited in the input table. Figure 2 shows this situation in a small workflow.

One of the important design decisions was to ensure easy extensibility, so that other users can add functionality, usually in the form of new nodes (and sometimes also data types). This has already been done by several commercial vendors (Tripos, Schrödinger, Symyx, ...) but also by other university groups or open source programmers. The usage of Eclipse as the core platform means that contributing nodes in the form of plugins is a very simple procedure. The official KNIME website offers several extension plugins for reporting via BIRT[2], statistical analysis with R[3] or extended machine learning capabilities from Weka[4], for example.

Since the initial release in mid 2006 the growing user base has had quite a lot of suggestions and request for improving KNIME's usability and functionality. Many of them were then integrated into the KNIME 2.0 version some of which we will describe in the following sections.

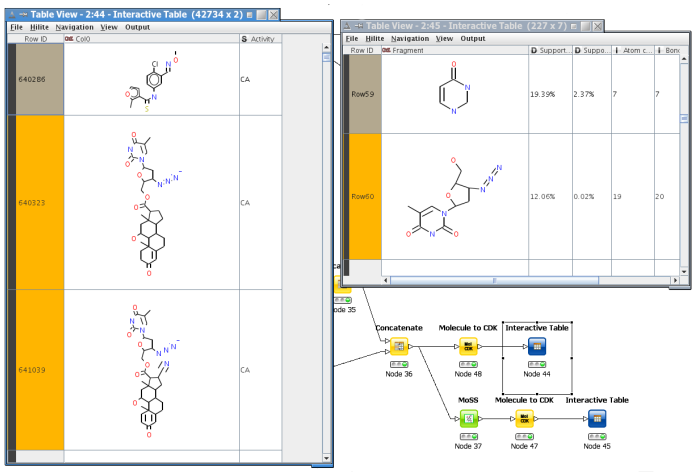

Fig. 2. KNIME's hiliting features demonstrated by the MoSS node for frequent fragment mining.

2 New features in 2.0

Besides several new nodes and a lot of work under the hood (see the KNIME website at http://www.knime.org/ for more details), we will discuss the following six new features in more detail: support for loops in the workflow, improved meta nodes, which encapsulate sub-workflows, a new concept of user-defined port objects in addition to data tables, improved database connectivity by using the new port objects, support of PMML in common data mining algorithms, and extended Weka integration.

2.1 Improved meta nodes

KNIME 1.x already featured the concept of meta nodes however they had a number of deficiencies. A meta node is a wrapper node that encapsulates parts of a workflow, be it for better structuring of the workflow or - as in KNIME 1.x - to implement special functionality such as cross validation (the latter has been replaced by the more powerful loop concept described below). Whereas previous KNIME versions had only a fixed set of meta nodes (1 or 2 data input/output ports), it is now possible to create meta nodes with an arbitrary number and even type (see section 2.3) of ports by using a simple wizard. These meta nodes can even be nested and copied.

2.2 Loop support

The workflows' conceptual structure is a directed acyclic graph, i.e. there are no loops from the output of one node to the input of one of its predecessors. Data

flows strictly in one direction. However, there are cases in which the repeated execution of parts of the workflow with changed parameters is desirable. This can range from simple iterations over several input files, to cross validation where a model is repeatedly trained and evaluated with different distinct parts of data, to even more complex tasks such as feature elimination. In order to be able to model such scenarios in KNIME, two special node types have been introduced: loop start- and loop end-nodes. In contrast to normal nodes (inside the loop) they are not reset while the loop executes, each of both nodes has access to its counterpart, and they can directly exchange information. For example, the loop end node can tell the start node which column it should remove at the next iteration or the start node can tell the end node if the current iteration is the last or not. Figure 3 shows a feature elimination loop in which the start and end nodes are visually distinguishable from normal nodes. The feature elimination model can then be used by the feature elimination filter to remove attributes from the data table.

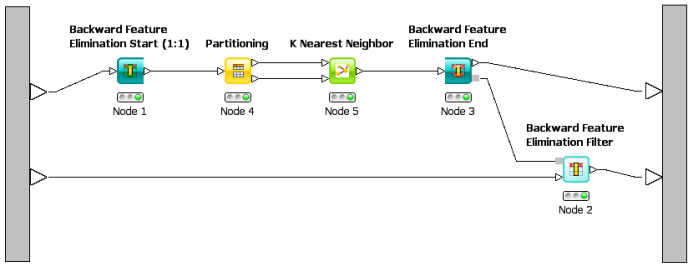

Fig. 3. The figure shows a feature elimination loop inside a meta node that iteratively removes one attribute after another, starting with the worst (i.e. whose removal degrade model quality the least).

KNIME 2.0 contains several pre-defined loops encapsulated in meta nodes:

- Simple "for" loop, which is executed a certain number of times
- Cross validation
- Iterative feature elimination
- Looping over a list of files

Programmers can easily write their own loop nodes by simply implementing an additional interface.

2.3 Port objects

In previous KNIME versions there were two types of ports, data ports and model ports. The latter did not distinguish between the actual type of models be it a decision tree, a neural net or even color information for data rows used in views. Therefore it was possible to connect a decision tree learner with a neural network

predictor and an error was only reported upon execution of the flow. From the programmer's point of view, in certain cases it was quite complicated to translate a model into the used data structure (nested key-value pairs) and back. KNIME 2.0 adds arbitrary port types that can easily be defined by the programmer. This has two excellent advantages: for the user it is now impossible to connect incompatible ports and the programmer is responsible for (de)serializing the transferred "port object" himself. This is usually much easier than using the old-style method and requires considerably less memory (and space on disk) for big models because the nested hash maps are omitted.

2.4 Improved database support

The new database ports are another good example of extended port object implementation. These database ports (dark red squares) pass on a connection that encapsulates the parameters used to establish a database connection via a JDBC-compliant bridge.

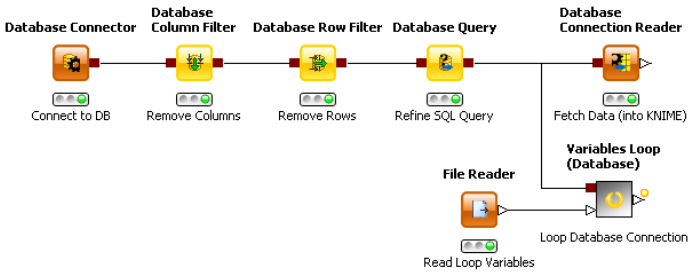

Fig. 4. Workflow with nodes that use the new database connections.

In the example above, the database nodes work directly in the database by modifying and wrapping SQL statements. The SQL statement itself is only executed when the data is imported into KNIME with the Database Connection Reader node (transition from database to data port). All other nodes, such as Database Connector, Database Column Filter, Database Row Filter and Database Query node perform well-defined operations on the SQL statement. In this example the database connection settings are adjusted within the Connector node and passed to the Database Column Filter and the Row Filter node. The filter nodes offer a user-friendly way to filter out columns and rows without modifying any SQL statement by hand. For advanced users, the SQL query node can be used to manually edit the SQL statement. The out-port view for each of those nodes supports a quick look into the database settings, the database meta data and – on explicit user request – the preview of the current data inside the database.

2.5 PMML

The Predictive Model Markup Language (PMML[5]) is an open standard for storing and exchanging predictive models such as cluster models, regression models, trees or support vector machines in XML format. Ideally, a model trained by SPSS for example, and stored as PMML can be used in R, SAS Enterprise Miner or, since version 2.0, also in KNIME. Almost all basic KNIME nodes that create a model output it in PMML (if the standard supports it). The corresponding predictor nodes take PMML as input. For PMML exchange PMML reader and writer nodes have been added. However, one should keep in mind, that the PMML standard offers many optional attributes in the model, which are usually only understood by the same application that created the model, meaning that in some cases interoperability is limited.

2.6 Extended Weka integration

KNIME 2.0 supports the new Weka version 3.5.6 [4]. Apart from the 50 classifiers that were already part of the Weka-Integration, meta-classifiers, cluster and association rule algorithms have also been integrated adding up to a total of approximately 100 Weka nodes in KNIME.

The new Weka port objects (see Section 2.3) are an important new feature. They enable a trained classifier or cluster model to be stored along with the used attributes and the target column. A view on this port lets the user explore the model or clustering that has been built with the training data. This model can be used to predict unseen data with the new Weka predictor node or to assign new data instances to clusters with the Weka cluster assigner node.

A preliminary attribute check in the dialog of all Weka nodes makes use of the Capabilites object in Weka. This allows the algorithm to be evaluated as to whether it can handle the input data, e.g. whether the algorithm supports numerical and/or nominal attributes.

References

1. Berthold, M.R., Cebron, N., Dill, F., Gabriel, T.R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Data Analysis, Machine Learning and Applications - Proceedings of the 31st Annual Conference of the Gesellschaft für Klassifikation e.V. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, Berlin, Germany (2007) 319–326
2. BIRT: Business Intelligence and Reporting Tools. http://www.eclipse.org/birt/
3. R Project: The R Project for Statistical Computing. http://www.r-project.org/
4. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2 edn. Morgan Kaufmann, San Francisco (June 2005)
5. Data Mining Group: Predictive Model Markup Language (PMML). http://www.dmg.org/