Scientific Workflow Management with ADAMS

Peter Reutemann[1] and Joaquin Vanschoren[2]

[1] University of Waikato, Hamilton, NZ
fracpete@waikato.ac.nz
[2] Leiden University, Leiden, NL
joaquin@liacs.nl

Abstract. We demonstrate the Advanced Data mining And Machine learning System (ADAMS), a novel workflow engine designed for rapid prototyping and maintenance of complex knowledge workflows. ADAMS does not require the user to manually connect inputs to outputs on a large canvas. It uses a compact workflow representation, *control operators*, and a simple interface between operators, allowing them to be auto-connected. It contains an extensive library of operators for various types of analysis, and a convenient plug-in architecture to easily add new ones.

Keywords: scientific workflows, machine learning, data mining.

1 Introduction

Many of today's data mining platforms offer *workflow engines* allowing the user to design and run knowledge workflows, from cleaning raw data to building models and making predictions. Most of these systems, such as Kepler [1], RapidMiner [2] and KNIME [3], represent these dependencies in a directed graph.[1] Many of them take a "canvas"-based approach, in which the user places operators on a large canvas and then connects the various inputs and outputs manually, thus introducing each dependency as a line on the canvas.

Though this is a very intuitive approach that greatly appeals to many end users, it is also a very time consuming one. When inserting additional operators, one has to move and rearrange the entire workflow to keep the design tidy. If an operator is replaced, all connections have to be redrawn. Moreover, scientific workflows often grow very complex, including hundreds of independent steps [5]. On a canvas, this leads to very large and complex graphs with many interconnections. This means that oversight is easily lost, even with useful features such as zooming, hierarchical workflows or meta-operators with internal workflows.

In this paper, we present ADAMS (Advanced Data mining And Machine learning System), a novel workflow engine specifically designed for rapid prototyping of complex scientific workflows, taking away the need to manually lay out and connect operators on a canvas. It presents the workflow in a compact tree structure in which operators can quickly be dragged in or pulled out, and

[1] For an in-depth overview and comparison of scientific workflow systems, see [4,5].

P. Flach et al. (Eds.): ECML PKDD 2012, Part II, LNCS 7524, pp. 833–837, 2012.

auto-connected to the surrounding operators. It includes an extensive library of operators, including a range of *control operators* to create and direct sub-flows. Moreover, new operators can be added very easily, either by dropping them in a folder, or writing them on-the-fly, without compilation, in scripts.

2 Workflow Representation

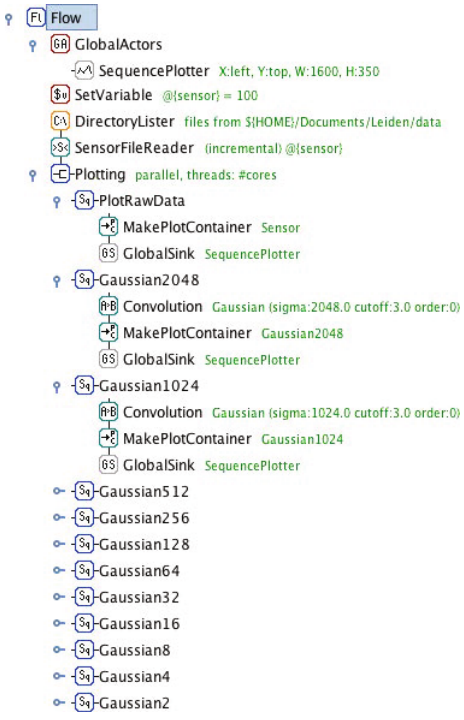

Fig. 1. Data visualization flow

Figure 1 illustrates an ADAMS workflow. It reads files from a directory with sensor data and plots the raw and convoluted data (scale-space composition). Operators (called 'actors') are dragged from a library into the flow, and will automatically 'snap' into the tree structure depending on where they are dropped. Actors are shown as nodes with a name and a list of parameter settings (options). They are color-coded based on whether they are a *source* (only output, e.g. `DirectoryLister`), *transformer* (input and output, e.g. `Convolution`), *sink* (only input, e.g. `SequencePlotter`) or *standalone* (no in/output, e.g. `GlobalActors`). Branches can be collapsed, and clicking actors opens a settings dialog. Some actors are fine-grained, allowing data to be manipulated within the flow, instead of requiring new actors.

Tokens. Data are passed as *tokens* wrapping a single Java object (e.g. a string or an entire dataset), as well as provenance information: a trace of actors affecting the data. Tokens can assume any level of granularity: actors can receive a single token (e.g., a dataset) and emit many (e.g., data points), or vice versa, buffer tokens and emit an array (e.g., the `SequenceToArray` actor). Actors with several outputs can attach a key to each token, creating key-value pairs. Such pairs can be combined in a *container*, and actors can extract tokens by key. For instance, `MakePlotContainer` attaches 'X' and 'Y' keys so that data can be plotted.

Control actors can branch or merge sub-flows and define how data is passed between their sub-actors. `Sequence` executes its sub-actors in sequence, passing tokens from one to the next. `Branch` forwards each received token to all underlying actors and executes them in parallel. `Tee` splits off each input

token, feeds it to its sub-flow and waits until it finishes before passing the token on. `Trigger` simply starts its sub-flow upon receiving a token (without feeding the token to its sub-flow). `Injector` passes each received token on, but also injects a new token. Some actors are conditioned on the value of the received token: `ConditionalTee` runs its sub-flow only if a stated condition holds, `If-Then-Else` runs one of two sub-workflows depending on a test, and `WhileLoop` loops over its sub-flow as long as its condition holds.

N-to-M semantics. While a tree representation cannot represent N-to-M relationships, ADAMS solves this shortcoming through *variables, key-value pairs, and global actors*. String tokens can be assigned to a variable using a `SetVariable` actor (e.g., `@{sensor}` in Fig.1), and used as an actor parameter or reintroduced elsewhere as a token by the `Variable` actor[2]. Similarly, any object token can be stored as a key-value pair by the `SetStorageValue` actor and reintroduced by the `StorageValue` actor. Finally, actors and their sub-flows can be made global: for instance, all tokens sent to a `GlobalSink` actor are passed to the referenced global actor, as shown in Fig. 1.

Interactivity. Actors can interact with the user when needed through dialog. For instance, they can ask the user to locate an undefined file, or display a number of results and allow the user to make a subselection before proceeding.

3 Plug-In Architecture

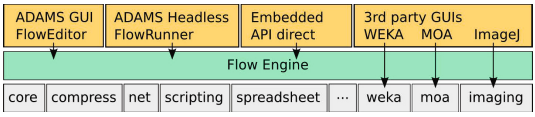

Fig. 2. ADAMS architecture

ADAMS contains an extensive library of actors enabling the inclusion of techniques from many existing libraries in a modular framework (see Fig.2). This includes actors for machine learning techniques, importing and exporting spreadsheets, generating graphics and PDF files and sending email. A concise overview of currently supported libraries is shown in Table 1. In addition, ADAMS has a plug-in architecture to easily add new actors. A new actor can be written as a single Java class implementing a simple API. When this file is dropped into a specific folder (icon optional), ADAMS will find it and show the actor in the workflow interface. Using one of the scripting languages, actors can be developed on-the-fly without compilation.

4 Applications

ADAMS is being used in two practical applications involving large, complex workflows. First, Gas Chromatography Mass Spectrometry (GC-MS) is a technique used to detect concentrations of compounds of interest, but the raw, high-dimensional data produced is generally not amenable to processing with machine

[2] The scope of variables can be limited to one specific sub-flow by a `LocalScope` actor.

Table 1. Overview of currently supported tools, available through actors

Task	Support for
Machine learning	WEKA, MOA, parameter optimization, experiment generation
Data Streams	MOA, Twitter
Spreadsheets	MS Excel, ODF, CSV
Graphics	BMP, JPG, PNG, TIF, PDF
Imaging	ImageJ, JAI, ImageMagick, Gnuplot
Scripting	Groovy, Jython
Other	HTTP, FTP, SFTP, SSH, Email, tar/zip/bzip2/gzip

learning systems. Using ADAMS, effective data flows were designed to entirely automate this process [6]. Second, in the InfraWatch project [7], a heterogeneous sensor network of over 150 sensors is monitoring the dynamic behavior and structural health of a highway bridge. The token-based design of ADAMS proved ideal for online processing of sensor data, and its quick workflow prototyping facilitates experimentation with novel time series analysis techniques.

5 Conclusions

Most scientific workflow engines use a canvas on which operators are manually arranged and connected. While this is certainly very intuitive and appealing for many end users, it is not ideal for handling very large, complex workflows. ADAMS is a rapid prototyping workflow engine designed for researchers and practitioners dealing with large workflows. It offers a wide range of operators, a plug-in architecture to include new ones on-the-fly, and a very compact workflow representation in which operators are auto-arranged, appreciatively speeding up workflow design and maintenance. Many examples and documentation can be found on ADAMS' website: `https://adams.cms.waikato.ac.nz/`

Acknowledgements. ADAMS was developed with funding from the New Zealand Ministry of Science and Innovation (MSI), contract UOWX0707. We also thank BiG Grid, the Dutch e-Science Grid and the Netherlands Organisation for Scientific Research, NWO.

References

1. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18, 1039–1065 (2006)
2. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2006)
3. 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, pp. 319–326 (2008)

4. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25, 528–540 (2009)
5. Bowers, S.: Scientific Workflow, Provenance, and Data Modeling Challenges and Approaches. Data Semantics 1, 19–30 (2012)
6. Holmes, G., Fletcher, D., Reutemann, P.: Predicting Polycyclic Aromatic Hydrocarbon Concentrations in Soil and Water Samples. In: Proceedings of the International Congress on Environmental Modelling and Software, IEMSS (2010)
7. Knobbe, A., Blockeel, H., Koopman, A., Calders, T., Obladen, B., Bosma, C., Galenkamp, H., Koenders, E., Kok, J.: InfraWatch: Data Management of Large Systems for Monitoring Infrastructural Performance. In: IDA Proceedings (2010)