Big Data with ADAMS

Peter Reutemann FRACPETE@WAIKATO.AC.NZ
Geoff Holmes GEOFF@WAIKATO.AC.NZ
Department of Computer Science
The University of Waikato
Hamilton, NZ

Abstract

ADAMS is a modular open-source Java framework for developing workflows available for academic research as well as commercial applications. It integrates data mining applications, like MOA, WEKA, MEKA and R, image and video processing and feature generation capabilities, spreadsheet and database access, visualizations, GIS, webservices and fast prototyping of new functionality using scripting languages (Groovy/Jython).

Keywords: workflow, big data, data streams, twitter, video, spectral data

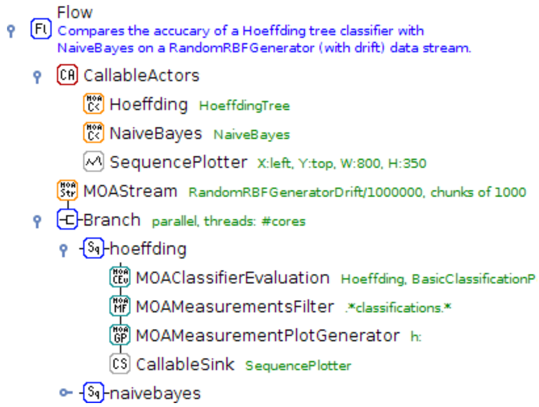
1. Introduction

The origins of ADAMS lie in academia, with the project originally being developed to process analytical data from gas-chromatography mass-spectrometry instruments (Holmes et al. (2010)), in order to handle many pre-processing steps and various predictions in parallel. However, the framework now forms the basis of commercial applications, as it allows for rapid development of data mining applications that integrate into business processes. The framework, written in Java and released under GPLv3 , consists of various modules grouped by functionality. It includes MOA (Bifet et al. (2010)), WEKA (Bouckaert et al. (2010)), MEKA (Read (2015)), R, image/video processing and feature generation, spreadsheet and database handling, various visualizations, GIS support through OpenStreetMap, webservices, and scripting (Jython or Groovy) for fast prototyping.

The remainder of the paper is structured as follows: first, a quick introduction to the framework is given. Second, an overview of some of the research areas that it can be used for. Third, a commercial application that is based on the framework.

2. Framework

Workflow systems, like RapidMiner, KNIME or Kepler, use a canvas approach for placing operators and combining them by connecting them. On the one hand, this is a very intuitive approach, since the user can see how the data flows. On the other hand, this can make modifying the flow a tedious process, as the authors experienced: disconnecting operators, moving them around, inserting other operators, reconnecting operators and so forth. Despite birds-eye-view and meta-operators that group several operators together, the canvas approach can quickly become hard to comprehend and doesn't seem to scale well when dealing with hundreds or even thousands of operators.



```
Flow
Compares the accucary of a Hoeffding tree classifier with
NaiveBayes on a RandomRBFGenerator (with drift) data stream.
  CallableActors
    Hoeffding  HoeffdingTree
    NaiveBayes  NaiveBayes
    SequencePlotter  X:left, Y:top, W:800, H:350
  MOAStream  RandomRBFGeneratorDrift/1000000, chunks of 1000
  Branch  parallel, threads: #cores
    hoeffding
      MOAClassifierEvaluation  Hoeffding, BasicClassificationP
      MOAMeasurementsFilter  .*classifications.*
      MOAMeasurementPlotGenerator  h:
      CallableSink  SequencePlotter
    naivebayes
```

Figure 1: Comparing two MOA classifiers.

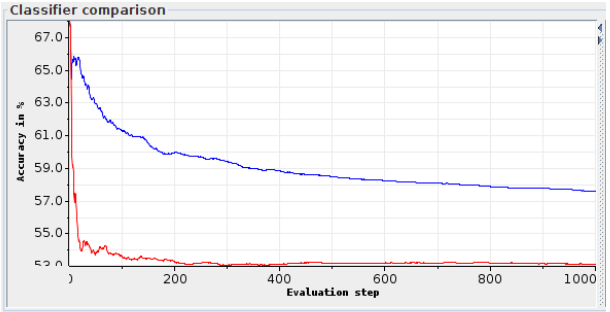


Figure 2: Comparison output.

For a simple streaming experiment that compares the accuracy of two classifiers (see Figure 1), the following steps would be set up in a flow: a stream generator that provides the data for training the classifiers; branching the data into two separate branches, each training and evaluating a classifier; the metric under investigation, i.e., accuracy, is extracted at specified intervals and turned into a plot data point; the plotting data from both classifiers is channelled into the same plot for direct comparison (see plot in Figure 2).

ADAMS forfeits some of the advantages and asthetics of this canvas approach to provide a more compact layout, using a simple tree structure instead for organizing its operators, which are called actors. The nesting of nodes in the tree resembles the nesting of actors programmatically, based on whether the actor is a primitive one (e.g., filtering a dataset) or one that manages other actors (e.g., grouping actors to be executed sequentially). So-called control actors determine data flow and flow execution: a Branch actor forwards the incoming data to all of its sub-actors, i.e., the sub-branches; a Sq Sequence actor executes its sub-actors one-by-one using the output of one actor as the input for the next one. The four types of combinations that result from input and output handling, are as follows: CA Standalone, which has no input nor output, e.g., a database connection; MOA Str Source, which only generates output; MOA CEv Transformer, which takes input and generates output; Sink, which only consumes data. By using data flow control actors, it is possible to get rid of explicit connections between the actors. However, a tree structure only handles 1-to-n connections. In order to simulate n-to-m semantics, several techniques are implemented: use of containers to combine multiple outputs; support for variables, to be attached to parameters of actors or simply used for calculations; internal key-value storage for storing, retrieving and updating data in multiple locations; and callable actors that can be invoked from anywhere in the flow using their name. Interoperability of actors is statically checked by comparing the types of data that a preceding actor can output and what types the following actor can handle. This type information in combination with the context induced by the tree structure, makes it possible to limit the actors that can be placed within a given context. Furthermore, context-aware rules can be specified for common sequences of actors (e.g., load dataset → set class attribute → cross-validate dataset → output results) aiding the user in creating flows more rapidly.

Adding new actors is straightforward: all it needs is a class either derived from one of the abstract superclasses or implementing the core interfaces for actors, placed in one of the packages which ADAMS inspects for actors and a little icon to be displayed in the editor.

3. Research

Given its origins, ADAMS is well-positioned for research. One of the standalone tools for stream analysis is MOA, which offers a user interface for running experiments, evaluating algorithms and visualizing metrics. However, in a workflow context, stream analysis becomes much more interesting, as filters can be appplied to the stream, compare several algorithms in parallel and plot their performance in a single plot (see Figures 1 and 2). Classification and regression experiments, as well as cluster visualizations are possible (see Figure 3).

Another area of research is Tweet analysis: tweets can be recorded and their associated meta-data using the public Twitter API, storing them for future replay. This tweet stream replay functionality allows the same experiment to be performed as often as required, using the same stream of tweets each time, applying different filters (e.g., checking for meta-data) and/or algorithms. Tweets with geotagging information can be displayed using the OpenStreetMap GIS functionality, allowing for visualization of geographical phenomena.

Due to a recently started collaboration with the Biology department, processing videos has been added to ADAMS. Videos get processed in near real-time, with frames being obtained at specific intervals. Apart from tracking objects, in our case tracking mice and rats during behavioural experiments (see Figure 4), it is also possible to use the image processing and feature generation functionality to generate input for machine learning algorithms, like MOA or WEKA.

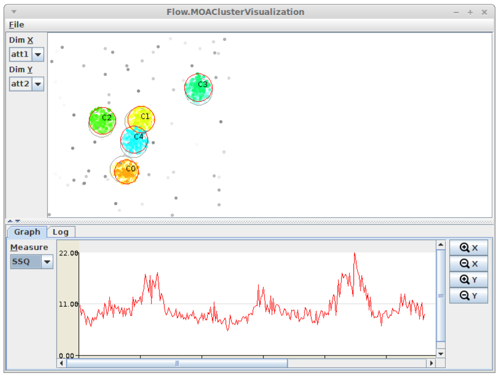


Figure 3: MOA cluster visualization.

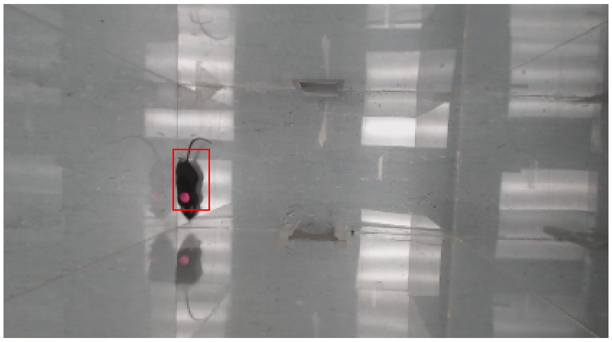


Figure 4: Tracking mice movements.

4. Industry

A commercial application based on the ADAMS framework is used for spectral analysis, encompassing near-infrared (NIR), mid-infrared (MIR) and X-ray fluorescence (XRF). One of our customers, BLGG AgroXpertus , is a large environmental laboratory in Europe with multiple sites across Europe. Traditionally, they analyzed plant and soil samples using only wet chemistry. This is a lengthy and costly process that can take up to a week and includes sample preparation, with toxic by-products, and the chemical analysis to determine concentrations of various compounds. However, when analyzing the samples in tandem with a rapid spectral technique, it is possible to build regression models with very good correlation and small errors. The number of data points per raw spectrum depends on the method and instrument, but is roughly 10,000 for XRF, 2,000 for MIR and 1,500 for NIR. The largest model for NIR is for soil and encompasses about 150,000 spectra. Altogether, BLGG uses around 250 different models for predicting various plant and soil properties. During normal operation, roughly 1,000 samples are processed each day, but during soil season (per legislation, every farmer in the Netherlands has to send in samples every four years) this goes up to 2-3,000 per day. By predicting the various concentrations using a rapid analysis technique like NIR, the number of samples that need to get analyzed via wet chemistry can be reduced significantly. The system has been in place now since 2006 and saves the company, at the time of writing, US$ 18 million per annum and US$ 33 million per annum during soil season. Using spectral analysis also enabled BLGG to scale up their operations. The use of machine learning here is a good repsonse to Kiri Wagstaff's *Impact Challenge #2: "$100M saved through improved decision making provided by an ML system"*, that she posed in 2012 for *machine learning that matters* (Wagstaff (2012)).

One difference to the default ADAMS framework is that this system uses a meta-flow approach. With these meta-flows, the customer only defines certain properties of a model, e.g., data cleaning, modelling parameters and evaluation, but not any of the low-level steps. Workflow generators then take these settings and turn them, on the fly, into actual worker flows (with thousands of actors) performing all the required data collection, processing and predictions steps for the different models.

Acknowledgments

The authors would like to thank the organizers, especially Albert Bifet, for inviting us to present our work.

References

Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive Online Analysis. *Journal of Machine Learning Research (JMLR)*, 2010.

Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. WEKA – Experiences with a Java Open-Source Project. *Journal of Machine Learning Research*, 11:2533–2541, 2010.

Geoffrey Holmes, Dale Fletcher, and Peter Reutemann. Predicting polycyclic aromatic hydrocarbon concentrations in soil and water samples. In *Proc International Congress on Environmental Modelling and Software*, 2010.

Jesse Read. MEKA: A Multi-label Extension to WEKA, 2015. URL http://meka.sourceforge.net/.

Kiri L. Wagstaff. Machine learning that matters. In *In Procs of ICML*, 2012.