MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering.

Albert Bifet ABIFET@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Department of Computer Science
University of Waikato, Hamilton, New Zealand

Philipp Kranen KRANEN@CS.RWTH-AACHEN.DE
Hardy Kremer KREMER@CS.RWTH-AACHEN.DE
Timm Jansen JANSEN@CS.RWTH-AACHEN.DE
Thomas Seidl SEIDL@CS.RWTH-AACHEN.DE
Data Management and Data Exploration Group
RWTH Aachen University, Germany

Editors: Tom Diethe, Nello Cristianini, John Shawe-Taylor

Abstract

Massive **O**nline **A**nalysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams. MOA is designed to deal with the challenging problem of scaling up the implementation of state of the art algorithms to real world dataset sizes. It contains collection of offline and online for both classification and clustering as well as tools for evaluation. In particular, for classification it implements boosting, bagging, and Hoeffding Trees, all with and without Naïve Bayes classifiers at the leaves. For clustering, it implements StreamKM++, CluStream, ClusTree, Den-Stream, D-Stream and CobWeb. Researchers benefit from MOA by getting insights into workings and problems of different approaches, practitioners can easily apply and compare several algorithms to real world data set and settings. MOA supports bi-directional interaction with WEKA, the Waikato Environment for Knowledge Analysis, and is released under the GNU GPL license.

1. Introduction

Nowadays, data is generated at an increasing rate from mobile applications, sensor applications, measurements in network monitoring and traffic management, log records or click-streams in web exploring, manufacturing processes, call detail records, email, blogging, twitter posts and others. In fact, all data generated can be considered as streaming data or as a snapshot of streaming data, since it is obtained from an interval of time.

In the data stream model, data arrive at high speed, and an algorithm must process them under very strict constraints of space and time. MOA is an open-source framework for dealing with massive, potentially infinite, evolving data streams.

A data stream environment has different requirements from the traditional batch learning setting. The most significant are the following:

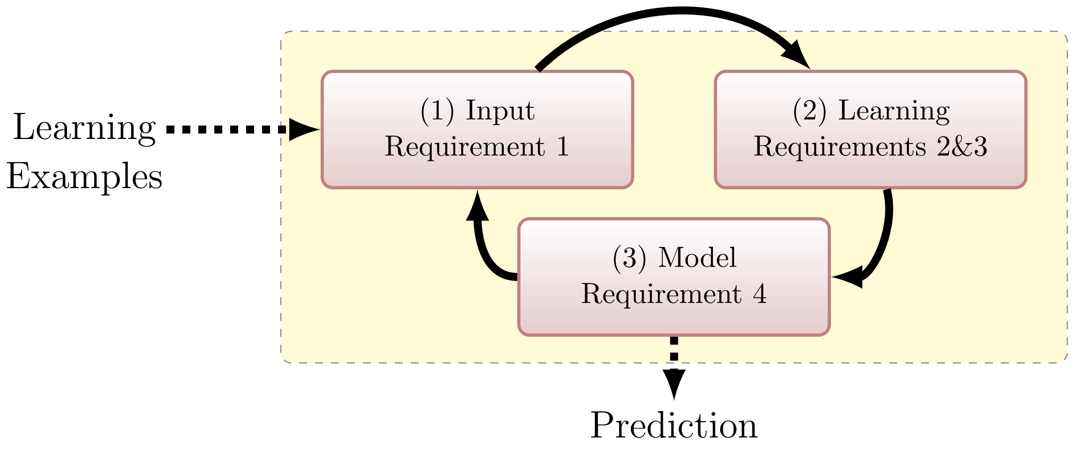


Figure 1: The data stream classification cycle

Requirement 1 Process an example at a time, and inspect it only once (at most)

Requirement 2 Use a limited amount of memory

Requirement 3 Work in a limited amount of time

Requirement 4 Be ready to predict at any time

Figure 1 illustrates the typical use of a data stream classification algorithm, and how the requirements fit in a repeating cycle:

1. The algorithm is passed the next available example from the stream (Requirement 1).

2. The algorithm processes the example, updating its data structures. It does so without exceeding the memory bounds set on it (requirement 2), and as quickly as possible (Requirement 3).

3. The algorithm is ready to accept the next example. On request it is able to predict the class of unseen examples (Requirement 4).

As data stream mining is a relatively new field, evaluation practices are not nearly as well researched and established as they are in the traditional batch setting. The majority of experimental evaluations use less than one million training examples. In the context of data streams this is disappointing, because to be truly useful at data stream classification the algorithms need to be capable of handling very large (potentially infinite) streams of examples. Demonstrating systems only on small amounts of data does not build a convincing case for capacity to solve more demanding data stream applications (Kirkby, 2007).

MOA permits evaluation of data stream learning algorithms on large streams, in the order of tens of millions of examples where possible, and under explicit memory limits. Any less than this does not actually test algorithms in a realistically challenging setting.

2. Classification

In traditional batch learning the problem of limited data is overcome by analyzing and averaging multiple models produced with different random arrangements of training and test data. In the stream setting the problem of (effectively) unlimited data poses different challenges. One solution involves taking snapshots at different times during the induction of a model to see how much the model improves.

The evaluation procedure of a learning algorithm determines which examples are used for training the algorithm, and which are used to test the model output by the algorithm.When considering what procedure to use in the data stream setting, one of the unique concerns is how to build a picture of accuracy over time. Two main approaches arise:

- **Holdout**: When traditional batch learning reaches a scale where cross-validation is too time consuming, it is often accepted to instead measure performance on a single holdout set. This is most useful when the division between train and test sets has been pre-defined, so that results from different studies can be directly compared.

- **Interleaved Test-Then-Train or Prequential**: Each individual example can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated. When intentionally performed in this order, the model is always being tested on examples it has not seen. This scheme has the advantage that no holdout set is needed for testing, making maximum use of the available data. It also ensures a smooth plot of accuracy over time, as each individual example will become increasingly less significant to the overall average (Gama et al., 2009).

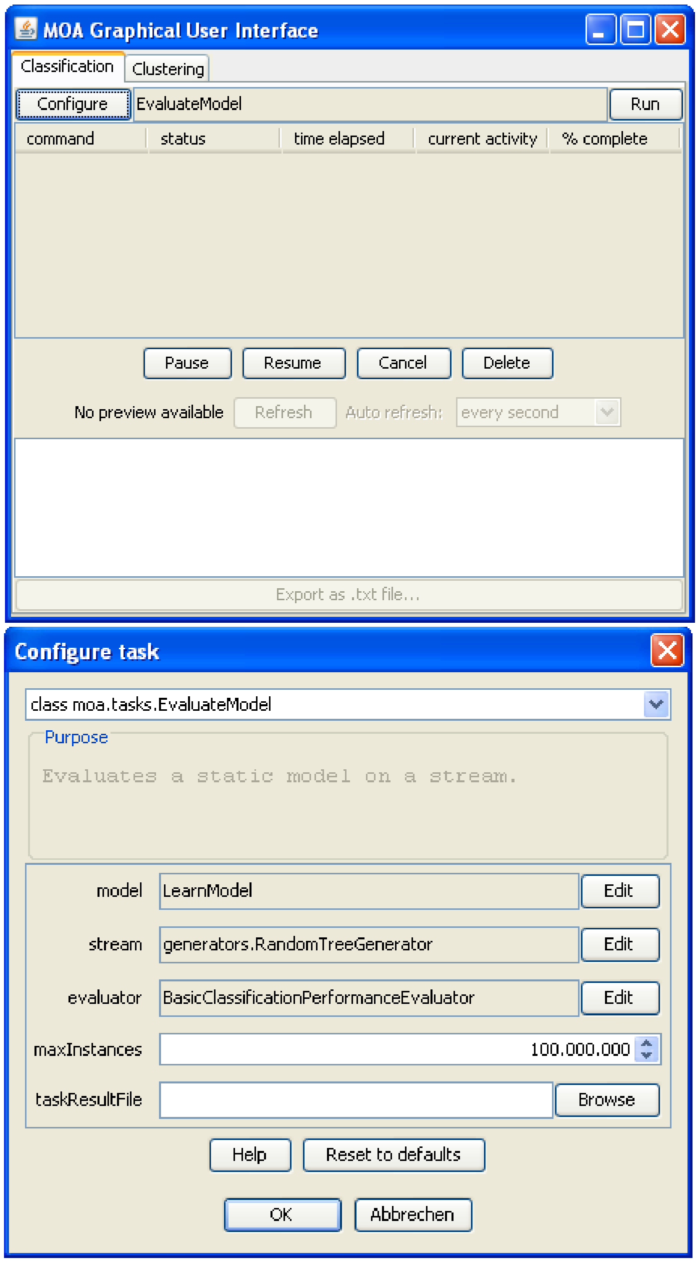
MOA contains stream generators, classifiers and evaluation methods. Figure 2 shows the MOA graphical user interface. However, a command line interface is also available.

Considering data streams as data generated from pure distributions, MOA models a concept drift event as a weighted combination of two pure distributions that characterizes the target concepts before and after the drift. Within the framework, it is possible to define the probability that instances of the stream belong to the new concept after the drift. It uses the sigmoid function, as an elegant and practical solution (Bifet et al., 2009a,b).

MOA contains the data generators most commonly found in the literature. MOA streams can be built using generators, reading ARFF files, joining several streams, or filtering streams. They allow for the simulation of a potentially infinite sequence of data. The following generators are currently available: Random Tree Generator, SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane, Random RBF Generator, LED Generator, Waveform Generator, and Function Generator. The implemented classifier methods currently include: Naive Bayes, Decision Stump, Hoeffding Tree, Hoeffding Option Tree (Pfahringer et al., 2008), Bagging, Boosting, Bagging using `ADWIN`, and Bagging using Adaptive-Size Hoeffding Trees (Bifet et al., 2009b).

A non-trivial example of the EvaluateInterleavedTestThenTrain task creating a comma separated values file, training the HoeffdingTree classifier on the WaveformGenerator data, training and testing on a total of 100 million examples, and testing every one million examples, is encapsulated by the following commandline:

```
java -cp .:moa.jar:weka.jar -javaagent:sizeofag.jar moa.DoTask \
```



EvaluateModel

class moa.tasks.EvaluateModel

Evaluates a static model on a stream.

model	LearnModel
stream	generators.RandomTreeGenerator
evaluator	BasicClassificationPerformanceEvaluator
maxInstances	100.000.000
taskResultFile	

Figure 2: MOA Graphical User Interface

```
"EvaluateInterleavedTestThenTrain -l HoeffdingTree \
-s generators.WaveformGenerator \
-i 100000000 -f 1000000" > htresult.csv
```

MOA is easy to use and extend. A simple approach to writing a new classifier is to extend `moa.classifiers.AbstractClassifier`, which will take care of certain details to ease the task.

3. Clustering

MOA contains also an experimental framework for clustering data streams, which allows comparing different approaches on individual (real world) settings and which makes it easy for researchers to run and build experimental data stream benchmarks. The features of MOA for stream clustering are:

- data generators for evolving data streams (including events such as novelty, merge, etc. (Spiliopoulou et al., 2006)),
- an extensible set of stream clustering algorithms,
- evaluation measures for stream clustering,
- visualization tools for analyzing results and comparing different settings.

For stream clustering we added new data generators that support the simulation of cluster evolution events such as merging or disappearing of clusters (Spiliopoulou et al., 2006). Currently MOA contains several stream clustering methods such as StreamKM++ (Ackermann et al., 2010), CluStream (Aggarwal et al., 2003), ClusTree (P. et al., 2010), Den-Stream (Cao et al., 2006), D-Stream (Tu and Chen, 2009) and CobWeb (Fisher, 1987). Moreover, MOA contains measures for analyzing the performance of the clustering models generated including measures commonly used in the literature as well as novel evaluation measures to compare and evaluate both online and offline components. The available measures evaluate both the correct assignment of examples (Chen, 2009) and the compactness of the resulting clustering.

Beside providing an evaluation framework, the second key objective is the extensibility of the benchmark suite regarding the set of implemented algorithms as well as the available data feeds and evaluation measures.

The visualization component allows to visualize the stream as well as the clustering results, choose dimensions for multi dimensional settings, and compare experiments with different settings in parallel. Figure 3 shows a screen shot of our visualization tab. For this screen shot two different settings of the CluStream algorithm (Aggarwal et al., 2003) were compared on the same stream setting (including merge/split events every 50000 examples) and five measures were chosen for online evaluation (CMD, F1, Precision, Recall and SSQ). The upper part of the GUI offers options to pause and resume the stream, adjust the visualization speed, choose the dimensions for x and y as well as the components to be displayed (points, micro- and macro clustering and ground truth). The lower part of the GUI displays the measured values for both settings as numbers (left side, including mean values) and the currently selected measure as a plot over the arrived examples (right, F1 measure in this example). For the given setting one can see a clear drop in the performance after the split event at roughly 160000 examples (event details are shown when choosing the corresponding vertical line in the plot). While this holds for both settings, the left configuration (red, CluStream with 100 micro clusters) is constantly outperformed by the right configuration (blue, CluStream with 20 micro clusters).

4. Website, Tutorials, and Documentation

MOA is a classification and clustering system for massive data streams with the following characteristics:

- open source tool and framework for practice, research and teaching similar to WEKA
- set of implemented algorithms for testing and comparison to approaches from the literature
- benchmark streaming data sets through stored, shared, and repeatable settings for the various data feeds and noise options, both synthetic and real

MOA is written in Java. The main benefits of Java are portability, where applications can be run on any platform with an appropriate Java virtual machine, and the strong and well-developed support libraries. Use of the language is widespread, and features such as the automatic garbage collection help to reduce programmer burden and error.

MOA can be found at:

`http://moa.cs.waikato.ac.nz/`

The website includes a tutorial, an API reference, a user manual, and a manual about mining data streams. Several examples of how the software can be used are available. Additional material regarding the extension of MOA to stream clustering can be found at

`http://dme.rwth-aachen.de/moa-datastream/`

The material includes a live video of the software as well as screenshots and explanations for the most important interfaces that are needed for extending our framework through novel data feeds, algorithms or measures.

5. Conclusions

Our goal is to build an experimental framework for classification and clustering on data streams similar to the WEKA framework. Our stream learning framework provides a set of data generators, algorithms and evaluation measures. Practitioners can benefit from this by comparing several algorithms in real world scenarios and choosing the best fitting solution. For researchers our framework yields insights into advantages and disadvantages of different approaches and allows the the creation of benchmark streaming data sets through stored, shared and repeatable settings for the data feeds. The sources are publicly available and are released under the GNU GPL license. Although the current focus in MOA is on classification and clustering, we plan to extend the framework to include regression, and frequent pattern learning (Bifet, 2010).

References

Marcel R. Ackermann, Christiane Lammersen, Marcus Märtens, Christoph Raupach, Christian Sohler, and Kamil Swierkot. StreamKM++: A clustering algorithm for data streams. In *SIAM ALENEX*, 2010.

Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

Albert Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams.* IOS Press, 2010.

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, and Ricard Gavaldà. Improving adaptive bagging methods for evolving data streams. In *First Asian Conference on Machine Learning, ACML 2009*, 2009a.

Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New ensemble methods for evolving data streams. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009b.

Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.

J. Chen. Adapting the right measures for k-means clustering. In *ACM KDD*, pages 877–884, 2009.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022852608280.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

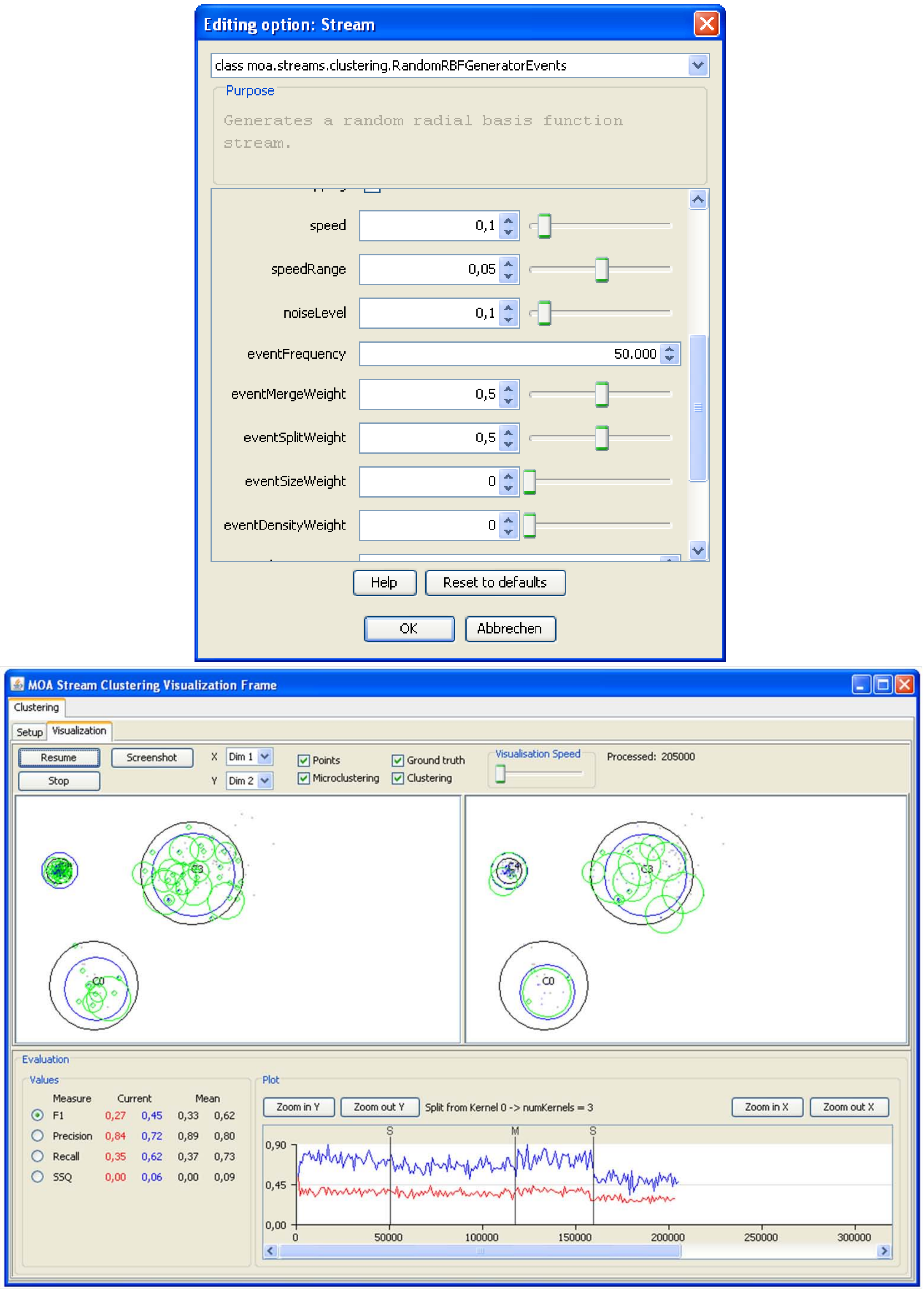
Richard Kirkby. *Improving Hoeffding Trees.* PhD thesis, University of Waikato, November 2007.

Kranen P., Assent I., Baldauf C., and Seidl T. The ClusTree: Indexing micro-clusters for anytime stream mining. In *Knowledge and Information Systems Journal (KAIS)*, 2010.

Bernhard Pfahringer, Geoff Holmes, and Richard Kirkby. Handling numeric attributes in hoeffding trees. In *PAKDD Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 296–307, 2008.

M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult. MONIC: modeling and monitoring cluster transitions. In *ACM KDD*, pages 706–711, 2006.

Li Tu and Yixin Chen. Stream data clustering based on grid density and attraction. *ACM Trans. Knowl. Discov. Data*, 3(3):1–27, 2009. ISSN 1556-4681. doi: http://doi.acm.org/10.1145/1552303.1552305.



class moa.streams.clustering.RandomRBFGeneratorEvents

Purpose

Generates a random radial basis function stream.

speed	0,1
speedRange	0,05
noiseLevel	0,1
eventFrequency	50.000
eventMergeWeight	0,5
eventSplitWeight	0,5
eventSizeWeight	0
eventDensityWeight	0

Processed: 205000

Values

Measure	Current		Mean	
F1	0,27	0,45	0,33	0,62
Precision	0,84	0,72	0,89	0,80
Recall	0,35	0,62	0,37	0,73
SSQ	0,00	0,06	0,00	0,09

Split from Kernel 0 -> numKernels = 3

Figure 3: Option dialog for the RBF data generator (by storing and loading settings benchmark streaming data sets can be shared for repeatability and comparison) and visualization tab of the clustering MOA graphical user interface.