2014 IEEE International Conference on Data Mining Workshop

# Big Data Stream Learning with SAMOA

Albert Bifet
HUAWEI Noah's Ark Lab, Hong Kong
bifet.albert@huawei.com

Gianmarco De Francisci Morales
Yahoo Labs, Barcelona
gdfm@yahoo-inc.com

***Abstract*—Big data is flowing into every area of our life, professional and personal. Big data is defined as datasets whose size is beyond the ability of typical software tools to capture, store, manage and analyze, due to the time and memory complexity. Velocity is one of the main properties of big data. In this demo, we present SAMOA (Scalable Advanced Massive Online Analysis), an open-source platform for mining big data streams. It provides a collection of distributed streaming algorithms for the most common data mining and machine learning tasks such as classification, clustering, and regression, as well as programming abstractions to develop new algorithms. It features a pluggable architecture that allows it to run on several distributed stream processing engines such as Storm, S4, and Samza. SAMOA is written in Java and is available at http://samoa-project.net under the Apache Software License version 2.0.**

## I. Introduction

Big data is "data whose characteristics forces us to look beyond the traditional methods that are prevalent at the time" [1]. Currently, there are two main ways to deal with big data: streaming algorithms and distributed computing (e.g., MapReduce). SAMOA[1] aims at satisfying the future needs for big data stream mining by combining the two approaches in a single platform under an open source umbrella [2].

Data mining and machine learning are well established techniques among web companies and startups. Spam detection, personalization and recommendation are just a few of the applications made possible by mining the huge quantity of data available nowadays.

The usual pipeline for mining and modeling data (what "data scientists" do) involves taking a sample from production data, cleaning and preprocessing it to make it amenable to modeling, training a model for the task at hand, and finally deploying it to production. The final output of this process is a pipeline that needs to run (and be maintained) periodically in order to keep the model up to date.

In order to cope with web-scale datasets, data scientists have resorted to *parallel and distributed computing*. MapReduce [3] is currently the de-facto standard programming paradigm in this area, mostly thanks to the popularity of Hadoop[2], an open source implementation of MapReduce started at Yahoo. Hadoop and its ecosystem (e.g., Mahout[3]) have proven to be an extremely successful platform to support the aforementioned process at web scale.

However, nowadays most data is generated in the form of a stream. Batch data is just a snapshot of streaming data obtained in an interval of time. Researchers have conceptualized and abstracted this setting in the *streaming model*. In this model data arrives at high speed, one instance at a time, and algorithms must process it in one pass under very strict constraints of space and time. Streaming algorithms make use of probabilistic data structures to give fast approximated answers.

On the one hand, MapReduce is not suitable to express streaming algorithms. On the other hand, traditional sequential online algorithms are limited by the memory and bandwidth of a single machine. *Distributed stream processing engines* (DSPEs) are a new emergent family of MapReduce-inspired technologies that address this issue. These engines allow to express parallel computation on streams, and combine the scalability of distributed processing with the efficiency of streaming algorithms. Examples of these engines include Storm[4], S4[5], and Samza[6].

[1] http://samoa-project.net

[2] http://hadoop.apache.org

[3] http://mahout.apache.org

[4] http://storm.apache.org

[5] http://incubator.apache.org/s4

[6] http://samza.incubator.apache.org

978-1-4799-4274-9/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDMW.2014.24

IEEE computer society

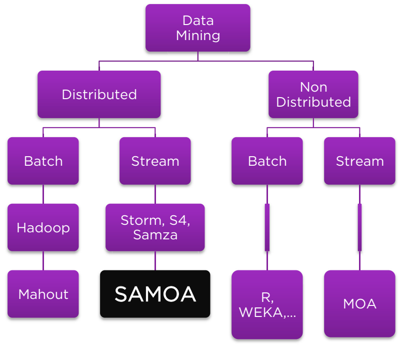

Fig. 1. Taxonomy of data mining and machine learning tools.

Alas, currently there is no common solution for mining big data streams, that is, for running data mining and machine learning algorithms on a distributed stream processing engine. The goal of SAMOA is to fill this gap, as exemplified by Figure 1.

## II. Description

SAMOA (Scalable Advanced Massive Online Analysis) is a platform for mining big data streams [4]. For a simple analogy, think of SAMOA as Mahout for streaming. As most of the rest of the big data ecosystem, it is written in Java.

SAMOA is both a framework and a library. As a framework, it allows the algorithm developer to abstract from the underlying execution engine, and therefore reuse their code on different engines. It features a pluggable architecture that allows it to run on several distributed stream processing engines such as Storm, S4, and Samza. This capability is achieved by designing a minimal API that captures the essence of modern DSPEs. This API also allows to easily write new bindings to port SAMOA to new execution engines. SAMOA takes care of hiding the differences of the underlying DSPEs in terms of API and deployment.

As a library, SAMOA contains implementations of state-of-the-art algorithms for distributed machine learning on streams. For classification, SAMOA provides a Vertical Hoeffding Tree (VHT), a distributed streaming version of a decision tree. For clustering, it includes an algorithm based on CluStream. For regression, HAMR, a distributed implementation of Adaptive Model Rules. The library also includes meta-algorithms such as bagging and boosting.

The platform is intended to be useful for both research and real world deployments.

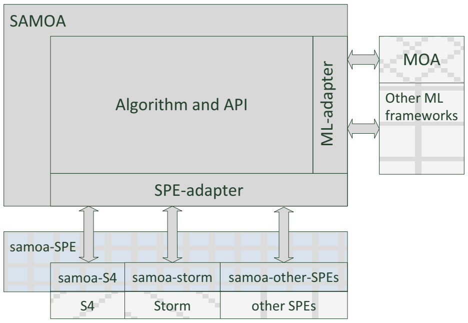

Fig. 2. High level architecture of SAMOA.

## III. High Level Architecture

We identify three types of SAMOA users:

1) Platform users, who use available ML algorithms without implementing new ones.
2) ML developers, who develop new ML algorithms on top of SAMOA and want to be isolated from changes in the underlying SPEs.
3) Platform developers, who extend SAMOA to integrate more DSPEs into SAMOA.

There are three important design goals of SAMOA:

1) **Flexibility** in term of developing new ML algorithms or reusing existing ML algorithms from other frameworks.
2) **Extensibility** in term of porting SAMOA to new DSPEs.
3) **Scalability** in term of handling ever increasing amount of data.

Figure 2 shows the high-level architecture of SAMOA which attempts to fulfill the aforementioned design goals. The *algorithm* layer contains existing distributed streaming algorithms that have been implemented in SAMOA. This layer enables platform users to easily use the existing algorithm on any DSPE of their choice.

The *application programming interface* (API) layer consists of primitives and components that facilitate ML developers when implementing new algorithms. The *ML-adapter* layer allows ML developers to integrate existing algorithms in MOA or other ML frameworks into SAMOA. The API layer and ML-adapter layer in SAMOA fulfill the flexibility goal since they allow ML developers to rapidly develop algorithms.

Next, the *DSPE-adapter* layer supports platform developers in integrating new DSPEs into SAMOA. To perform the integration, platform developers should implement the *samoa-SPE* layer as shown in Figure 2. Currently SAMOA is equipped with three adapters: the *samoa-Storm* adapter for Storm, the *samoa-S4* adapter for S4, and the *samoa-Samza* adapter for Samza. To satisfy the extensibility goal, the DSPE-adapter layer decouples DSPEs and ML algorithms implementations in SAMOA, so that platform developers are able to easily integrate more DSPEs.

The last goal, scalability, implies that SAMOA should be able to scale to cope ever increasing amount of data. To fulfill this goal, SAMOA utilizes modern DSPEs to execute its ML algorithms. The reason for using modern DSPEs such as Storm, S4, and Samza in SAMOA is that they are designed to provide horizontal scalability to cope with web-scale streams.

## IV. System Design

An algorithm in SAMOA is represented by a directed graph of nodes that communicate via messages along streams which connect pairs of nodes. Borrowing the terminology from Storm, this graph is called a *Topology*. Each node in a Topology is a *Processor* that sends messages through a *Stream*. A Processor is a container for the code implementing the algorithm. A Stream can have a single source but several destinations (akin to a pub-sub system). A Topology is built by using a *TopologyBuilder*, which connects the various pieces of user code to the platform code and performs the necessary bookkeeping in the background. The following is a code snippet to build a topology that joins two data streams in SAMOA:

```
TopologyBuilder builder = new
    TopologyBuilder();
Processor sourceOne = new SourceProcessor();
builder.addProcessor(sourceOne);
Stream streamOne = builder
     .createStream(sourceOne);

Processor sourceTwo = new SourceProcessor();
builder.addProcessor(sourceTwo);
Stream streamTwo = builder
     .createStream(sourceTwo);

Processor join = new JoinProcessor();
builder.addProcessor(join)
     .connectInputShuffle(streamOne)
     .connectInputKey(streamTwo);
```

A *Task* is an execution entity, similar to a job in Hadoop. A Topology is instantiated inside a Task to be run by SAMOA. An example of a Task is *PrequentialEvaluation*, a classification task where each instance is used for testing first, and then for training.

A message or an event is called *Content Event* in SAMOA. As the name suggests, it is an event which contains content which needs to be processed by the processors. Finally, a *Processing Item* is a hidden physical unit of the topology and is just a wrapper of Processor. It is used internally, and it is not accessible from the API.

## V. Machine Learning Algorithms

The Vertical Hoeffding Tree (VHT) is a distributed extension of the VFDT [5]. The VHT uses vertical parallelism to split the workload across several machines. Vertical parallelism leverages the parallelism across attributes in the same example, rather than across different examples in the stream. In practice, each training example is routed through the tree model to a leaf. There, the example is split into its constituting attributes, and each attribute is sent to a different Processor instance that keeps track of sufficient statistics. This architecture has two main advantages over one based on horizontal parallelism. First, attribute counters are not replicated across several machines, thus reducing the memory footprint. Second, the computation of the fitness of an attribute for a split decision (via, e.g., entropy or information gain) can be performed in parallel. The drawback is that in order to get good performances, there must be sufficient inherent parallelism in the data. That is, the VHT works best for sparse data.

SAMOA includes a distributed version of CluStream, an algorithm for clustering evolving data streams. CluStream keeps a small summary of the data received so far by computing micro-clusters online. These micro-clusters are further refined to create macro-clusters by a micro-batch process, which is triggered periodically. The period is configured via a command line parameter (e.g., every 10 000 examples).

For regression, SAMOA provides a distributed implementation of Adaptive Model Rules [6]. The algorithm, HAMR, uses a hybrid of vertical and horizontal parallelism to distribute AMRules on a cluster.

SAMOA also includes adaptive implementations of ensemble methods such as bagging and boosting.

These methods include state-of-the-art change detectors such as as ADWIN, DDM, EDDM, and Page-Hinckley [7]. These meta-algorithms are most useful in conjunction with external single-machine classifiers which can be plugged in SAMOA. For instance, open-source connectors for MOA [8] are provided separately by the SAMOA-MOA package[7].

The following listing shows how to download, build and run SAMOA.

```
# download and build SAMOA
git clone git@github.com:yahoo/samoa.git
cd samoa
mvn package

# download the Forest Cover Type dataset
wget "http://downloads.sourceforge.net/
    project/moa-datastream/Datasets/
    Classification/covtypeNorm.arff.zip"
unzip "covtypeNorm.arff.zip"

# run SAMOA in local mode
bin/samoa local target/SAMOA-Local-0.0.1-
    SNAPSHOT.jar "PrequentialEvaluation -l
    classifiers.ensemble.Bagging -s (
    ArffFileStream -f covtypeNorm.arff) -f
    100000"
```

## VI. Demonstration Plan

We will showcase our prototype system and demonstrate its usage, user interaction, fast methods, and scalable system design through different scenarios on several data streams.

We would like to discuss with experts and practitioners to get new input and ideas for further features and extensions of our framework. Particularly, we would also like to focus on giving insight into workings and drawbacks of different approaches, for example, in which situation does an algorithm or an evaluation measure fail.

The audience will benefit from knowing that a distributed learning system for massive data streams with the following characteristics is available:

- benchmark settings for streaming data through shared and repeatable settings;
- set of implemented algorithms for comparison to approaches from the literature;
- set of evaluation measures and methods to assess their performance on evolving data streams;
- open source license useful for research, teaching, and real-world deployments.

## VII. Conclusions

This demonstration builds upon the SAMOA platform for mining big data streams. The platform supports the most common machine learning tasks such as classification, clustering, and regression. It also provides a simplified API for developers that allows to implement distributed streaming algorithms easily.

SAMOA is already available and can be found online at http://www.samoa-project.net. The website includes a wiki, an API reference, and a developer's manual. Several examples of how the software can be used are also available. The code is hosted on GitHub. SAMOA contains a test suite that is run on each commit on the GitHub repository via a continuous integration server[8]. Finally, SAMOA is released as open source software under the Apache Software License v2.0.

## References

[1] A. Jacobs, "The Pathologies of Big Data," *Communications of the ACM*, vol. 52, no. 8, pp. 36–44, Aug. 2009.

[2] G. De Francisci Morales and A. Bifet, "SAMOA: Scalable Advanced Massive Online Analysis," *JMLR: Journal of Machine Learning Research*, 2014.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data processing on Large Clusters," in *OSDI '04: 6th Symposium on Opearting Systems Design and Implementation*. USENIX Association, 2004, pp. 137–150.

[4] G. De Francisci Morales, "SAMOA: A Platform for Mining Big Data Streams," in *RAMSS'13: 2nd International Workshop on Real-Time Analysis and Mining of Social Streams @WWW'13*, 2013.

[5] P. Domingos and G. Hulten, "Mining high-speed data streams," in *KDD '00: 6th international conference on Knowledge Discovery and Data mining*, 2000, pp. 71–80.

[6] A. Thu Vu, G. De Francisci Morales, J. Gama, and A. Bifet, "Distributed Adaptive Model Rules for Mining Big Data Streams," in *BigData '14: Second IEEE International Conference on Big Data*, 2014.

[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Computing Surveys*, vol. 46, no. 4, 2014.

[8] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[7] https://github.com/samoa-moa/samoa-moa

[8] https://travis-ci.org/yahoo/samoa