International Journal of Database Theory and Application
Vol.7, No.4 (2014), pp. 197-208
http://dx.doi.org/10.14257/ijdta.2014.7.4.15

Handling Big Data Stream Analytics using SAMOA Framework - A Practical Experience

Bakshi Rohit Prasad and Sonali Agarwal

Indian Institute of Information Technology, Allahabad
rohit.cs12@gmail.com, sonali@iiita.ac.in

Abstract

Data analytics and machine learning has always been of great importance in almost every field especially in business decision making and strategy building, in healthcare domain, in text mining and pattern identification on the web, in meteorological department, etc. The daily exponential growth of data today has shifted the normal data analytics to new paradigm of Big Data Analytics and Big Data Machine Learning. We need tools to perform online data analysis on streaming data for achieving faster learning and faster response in data analytics as well as maintaining scalability in terms of huge volume of data. SAMOA (Scalable Advanced Massive Online Analysis) is a recent framework in this reference. This paper discusses the architecture of this SAMOA framework and its directory structure. Also it expresses a practical experience of configuring and deployment of the tool for handling massive online analysis on Big Data.

Keywords: *Big Data; SAMOA; Stream Data; Stream Data Analytics; Massive Online Analysis; Distributed Framework; Machine Learning*

1. Introduction

Analyzing the data for extracting new patterns and knowledge is not a new field. Machine learning and data mining is being done for several decades for the sake of knowledge discovery and understanding the underlying patterns, hidden in the data [1]. This knowledge puts the basic building block in decision support systems and in strategy making in several domains such as in healthcare systems to diagnose disease patterns [2] of several diseases such as diabetes [3, 4], heart disease [5], cancer [6] *etc*, suggesting appropriate therapies, *etc.*, in business for identifying business trends as well as customer behavior and making strategies accordingly, in fraud and crime detection, in mining the web which could be a potential source of knowledge regarding user behavior for identifying threat patterns, opinion mining, sentiment analysis, *etc*.

But the way, the data has shown the tremendous growth with a rapid speed, the traditional computing techniques are becoming incapable of handling such Big Data [7] because of the following nature of Big Data [8, 9]:

i. Volume: Limitation of amount of memory available for computation is quite incapable of handling huge data.

ii. Variety: Data coming from various sources is of different type. It may be structured such as relational database, semi-structured such as XML documents or unstructured such as audio, video, text, *etc*. Hence devising algorithm to incorporate handling such variety of data is a complex task to achieve.

iii. Velocity: The rapid rate of data generation throws the algorithms and techniques out of the box, when used for the real time applications that require very fast computation to achieve faster response time.

Stream analytics focuses on the velocity characteristic of the Big Data. It analyzes the sequence of data (stream) online. Now there are two possibilities for performing stream analysis. One kind of processing is non distributed processing framework such as MOA (Massive Online Analysis) framework [8] where the stream processing task is handled by a single machine that takes ample amount of time in processing and generating the response. Also it is not that capable of handling Big Data which is tremendously huge. The other kind of processing may be distributed processing which distributes the processing among a cluster of processing nodes that easily handle huge data, parallelize the tasks and speed up the processing as well. SAMOA (Scalable Advanced Massive Online Analysis) is a recent framework that supports scalability, volume and velocity simultaneously. It is java based open source framework licensed to Apache, designed for distributed machine learning. It allows several distributed processing platforms to plug into the framework easily.

The paper is woven in four sections. The first section gives a brief introduction about the problems in handling Big Data and the requirement of distributed scalable processing computation framework for online stream data mining and machine learning. In the second section, the background of stream analysis and distributed stream data mining has been discussed. The third section explains the architecture and modular components of SAMOA framework. Then it gives a practical use case of SAMOA installation and configuration in detail to execute an example task in SAMOA.

2. Background

Machine learning is not a new field a lot of learning algorithms in terms of classification and clustering has been devised in previous decades. Initially they used to work in single machine environment. But the need for faster computing without having a high end computing machine gave birth to distributed computing and frameworks. Examples of currently available distributed machine learning frameworks are Apache Mahout [11] and ML-Base [12]. These frameworks deal with large volumes effectively and in variety of data dimensions. Mahout algorithms are implemented to run on the top of Hadoop, hence they provide batch kind of stream processing. The use of MLBase facilitates the easy application algorithms on top of distributed computing environment.

The data stream mining is recent attraction of research in field of data mining and machine learning. In this context, a machine learning algorithm is fruitful only if it can handle very large data streams [13]. Massive Online Analysis (MOA) framework can work on streams of order of tens of millions. But MOA processing is bounded by the memory limits of the system [10]. To overcome this problem the concept of distributed machine learning on stream data evolved. MOA [10, 14] and Vowpal Wabbit [15] are the machine learning frameworks that are created to solve the velocity problem of Big Data and capable of handling variety of data dimensions. They also provided APIs so that new machine learning algorithm can be developed on top of them.

Recent trends focused on the solutions that are capable to handle all the three big data challenges of volume, velocity and variety while doing stream mining. Most solutions existing today, only solves at most two challenges. SAMOA is the recent and open source framework for distributed machine learning that addresses all the three challenges while doing Big Data stream mining. The position of SAMOA in the hierarchy of Big Data computing is given in Figure 1.

The detail description of SAMOA and its practical experience in executing a task is given in subsequent sections. But, for smooth understanding of the working of SAMOA, we need to be familiar with some of the important entities involved in installing and configuring SAMOA. They are Maven and GitHub described subsequently.

2.1. Maven Building Tool

Maven [16] is a building tool written in java to develop Java software projects very easily. Though these projects may be written in other languages and may be built, but java is preferred. A build tool automates the process of building a software project which reduces chances of human errors in doing it manually and automated process is faster too. The software project building may have these activities:

- Source code generation (if project has provision for auto-generated code).
- Creating documentation using the underlying source code.
- Source code compilation.
- Creating archives such as JAR, ZIP files that contain the compiled code.
- Installing this packaged JAR in some repository locally or on server.

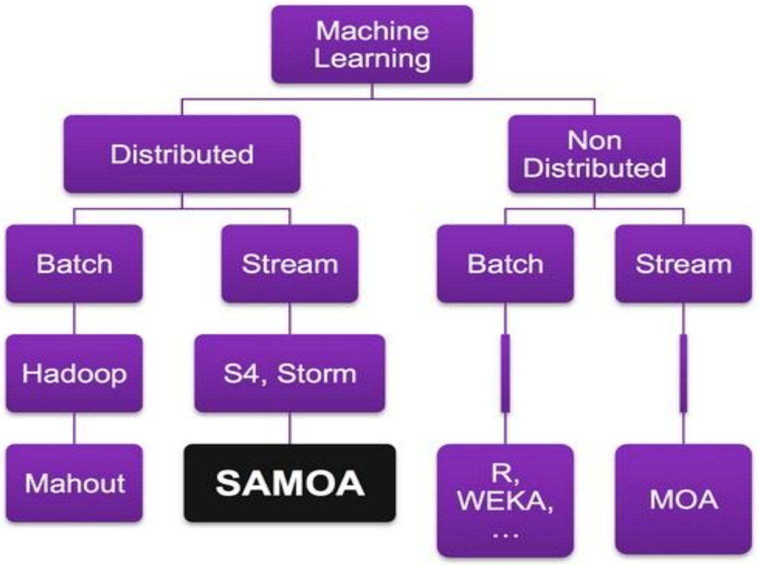


Figure 1. Position of SAMOA in Big Data Computing Hierarchy [20]

2.2. Standard Directory Structure of Maven

The standard structure of directory hierarchy for any Maven project is as depicted in Figure 2. The root directory is *'src'* directory that contains source code as well as test code. All the source codes excepting the test code lie inside the *'main*' directory. The test source code resides inside the *'test'* directory.

Application specific Java source code is stored in *'java'* sub-directory of main directory whereas the corresponding the Java code for testing lies in *'java'* sub-directory of test directory. Other many resources, such as property files of an application, used by the project are contained in the 'resources' directory.

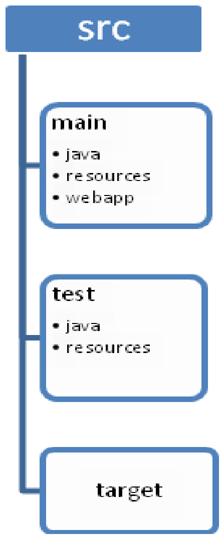


Figure 2. Standard Directory Structure of Maven

2.3. GitHub

GitHub [17] is hosting service on web, used for software projects development for the projects using GitHub revision control facility. GitHub comes with both paid plans and free accounts. Paid plans are required for private repositories whereas for open source projects free account is suitable. A user profile is required before using it. It facilitates live conversations so that people may discuss about code and manage or review changes to it. It also facilitates code sharing with team members, friends, or someone else. One can revise the projects by making changes to others' work and interact with team workers. In this way, it allows team workers to contribute to the same documents since a version control application enable several people to work simultaneously on same document without risk overwriting or erasing the work.

3. SAMOA Framework

SAMOA is an open source distributed framework that encompasses machine learning on streaming data in distributed fashion. SAMOA allows easy plug-in of various stream processing platforms within it. It contains distributed algorithms for classification and clustering, the most common machine learning tasks [18].

3.1. SAMOA Users and Design Goals

There are three types of users and three identified design goals of SAMOA as depicted in Figure 3 with an interrelation mapping between users and design goals.

3.1.1. SAMOA Users: The SAMOA users are one of the following types:

Platform users: These users intend to use the existing ML Algorithms only to handle Big Data.

ML developers: These kinds of users intend to reuse the available algorithms to develop new ones.

Platform developers: These users try to develop and plug in new distributed stream processing platform to extend SAMOA.

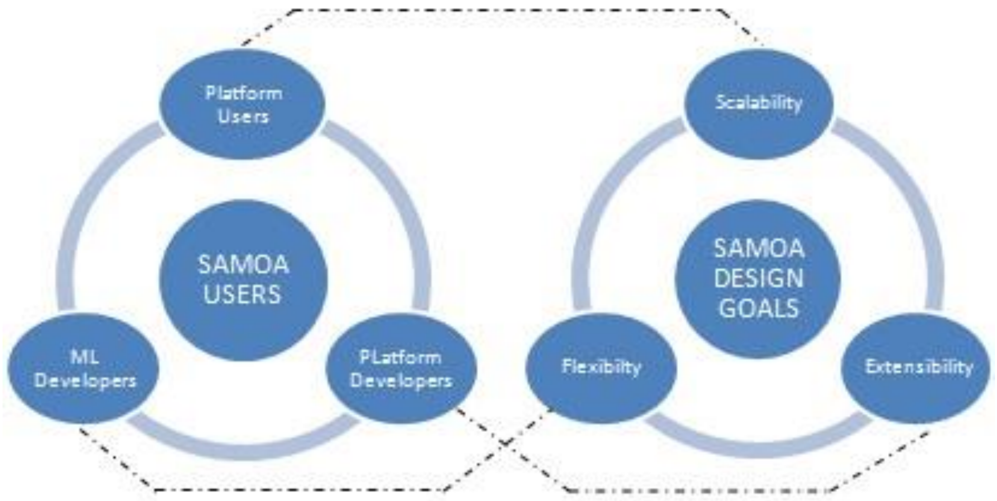


Figure 3. SAMOA Users and Design Goals Interrelation

3.1.2. SAMOA Design Goals: SAMOA design goals focus on following aspects:

Flexibility: This goal achieves development of new ML algorithms. Also, it allows reusing the already available algorithms that exist in other ML-frameworks.

Extensibility: This goal achieves development and integration of new stream processing platforms to extend the SAMOA framework.

Scalability: Allows enable platform users to handle rapidly growing amount of data in effective way.

3.2. Usage Perspective Architecture of SAMOA

SAMOA can be used from two perspectives as mentioned below:

3.2.1. Machine Learning: The existing machine learning tasks in SAMOA can be directly used or developers can implement new algorithms from scratch or by reusing existing one and run them.

3.2.2. Stream Processing Platform Abstraction: In this perspective new platforms can be developed and added by using the available API in SAMOA.

In this way the entire SAMOA project is bifurcated in two parts: SAMOA-API and SAMOA-Platform. Each part provides certain abstraction for developers. The SAMOA-API allows development of new algorithms without concerning about the underlying platform. The SAMOA-Platform facilitates adding up new distributed processing platforms without concerning about the existing APIs. The first as well as current release of SAMOA incorporates only two platforms; one is Apache S4 [19] and other is Twitter Storm. The architecture of the SAMOA in Figure 4 very clearly depicts the separation of roles mentioned above.

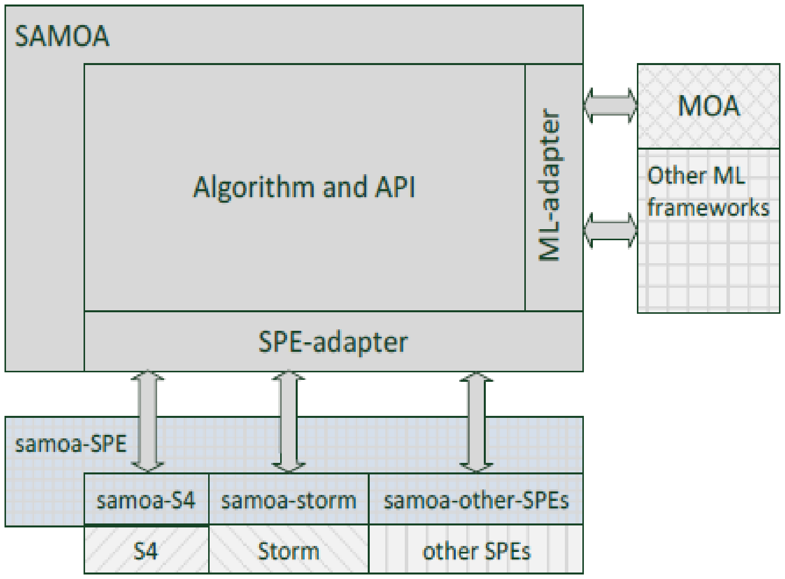


Figure 4. High Level Architecture of SAMOA [18]

As shown in Figure 4, the available distributed streaming algorithms in SAMOA lie in the algorithm block of the architecture enabling the platform users to easily use them while abstracting complexity of the underlying processing platform. The API block represents those components using which new algorithms can be developed by ML developers.

The ML-adapter layer facilitates integration of already existing algorithms in other ML-frameworks such as MOA, *etc*. The combined package <API, ML-adapter layer> achieves flexibility goal. Platform developers implement the samoa-SPE layer and use SPE-adapter layer to integrate new SPEs (Stream Processing Engines) into SAMOA. The current SAMOA contains only samoa-S4 layer and samoa-Storm layer corresponding to S4 and Storm respectively. SPE-Adapter layer provides the required abstraction from ML algorithm implementation and achieves the goal of extensibility. However, the scalability goal is achieved by using the most recent SPEs like Storm and S4 that are designed to effectively scale horizontally to handle continuously growing data.

3.3. SAMOA Modular Components

SAMOA APIs and modular components are reusable components that provide an effective tool to ML developers so that they may rapidly develop new algorithms. Following components have been shipped with SAMOA framework:

3.3.1. Processor: In SAMOA is reusable basic unit of computation. Each processor is nothing but actual logic that is responsible for executing some part of the algorithm corresponding to a specific processing platform. This logic is written by ML developers.

3.3.2. Processing Items (PI): PIs provide concrete implementation of reusable processor components. Figure 5 shows how the same processor is wrapped for a different processing platforms and form different processing items for various platforms.

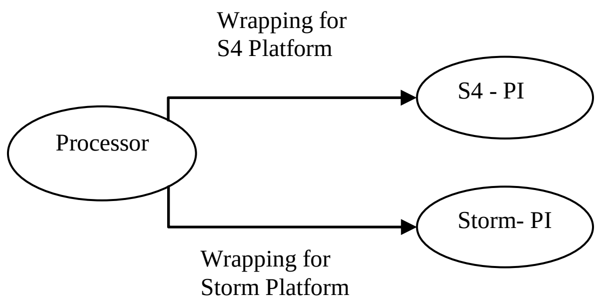


Figure 5. Reuse of Processor for Generation of New Processing Items

The SAMOA processing items are categorized in two types; Entrance PI and Simple PI. The entrance PI is responsible for collecting the data from external sources and converting it into instances or itself generating instances. These instances are sent to other one or more PI. The simple PI receives these content events and consumes them. The task of instantiation of PIs is taken care of SPE-Adapter layer. The instantiation is dynamically done by SAMOA according to underlying SPE.

3.3.3. Stream and Content Event: A stream connects two PIs; from source PI to destination PIs. It is the stream which carries content event between PIs. It is implemented as Java Interface. In SAMOA, a content event is a reusable component that can be considered as wrapping of the data transmitted between PIs. The instances to be transmitted are wrapped inside a content event and sent via a stream. The simple PI receives these content events and consumes them. A stream is instantiated by ML developers when they associate the stream to specific source PI. Content events are also designed as Java interface. It is the ML developers' task to give its concrete implementation.

3.3.4. Topology: A topology represents an organization of various components (processing items and streams) connected to each other to process the incoming data streams in desired way. A distributed streaming ML algorithm in SAMOA simply corresponds to a topology.

3.3.5. Task: Any activity related to machine learning such as evaluation of a specific classifier, is a task in SAMOA. For example, the PrequentialEvaluation task is a suitable example. This task receives an instance, test the performance of a model (classifier) using the instance and subsequently trains the model based on the same instance. A topology in SAMOA simply executes a task. The topology is automatically formed corresponding to a task.

Platform users are required to configure SAMOA for the available SPE for deployment. After successful configuration, the topology corresponding to the SPE is deployed and users can easily execute the task on configured cluster.

3.3.6. ML-adapter Layer: This layer has classes that perform wrapping of ML algorithms present in other ML frameworks. Current release of SAMOA wraps only MOA algorithms. The integration of this layer with other Java-based ML frameworks is very easy as SAMOA has java implementation. Non-Java ML frameworks can also be integrated via Java utilities such as JNI.

4. Execution of SAMOA Sample Example from the Scratch

4.1. Installing and Configuring Maven

The local computer installation of Maven requires following steps:

1. Download (from maven download page) and unzip the Maven in desirable directory.

2. M2_HOME environment variable is assigned the path of unzipped Maven directory.

3. M2 environment variable is assigned the path of M2_HOME/bin as below:
   - On Windows Machine- *M2=%M2_HOME%\bin*
   - On Unix Machine- *M2=$M2_HOME/bin*

4. Set PATH environment variable to point M2 as below:
   - On Windows Machine- *PATH=%M2%*
   - On Unix Machine- *PATH=$M2*

5. Check whether Maven has been successfully configured or not: Run *mvn* command on command prompt. If *mvn* command is recognized, that means Maven is successfully installed.

4.1.1. Maven Settings File: The settings files in Maven are used to configure Maven settings across all POM files. For example, one can configure, Active build profile, proxy, local repository location and many more. The Maven may optionally have two setting files named settings.xml in following locations:

- The root directory where Maven is installed:
  $M2_HOME/conf/settings.xml

- The home directory of the user:
  ${user.home}/.m2/settings.xml

If both settings file are present in system, then the values of user settings file overrides the values of settings file in Maven installation directory.

4.2. Download SAMOA

Clone SAMOA project source files from GitHub using following command:

```
git  clone https://github.com/yahoo/samoa.git

cd  samoa
```

4.3. Execute the Package Build Phase

It packs the compiled code of SAMOA in a distributable format, such as a JAR, using following command:

```
mvn  package
```

4.4. Collect Data Set

Here, taken covtypeNorm.arff data set which is a US Forest Research data set. The following command can executed to download the data set and unzipping it or it can manually downloaded from the given link in the command and unzip it into the SAMOA installation directory.

4.4.1. Command for Downloading Dataset

```
wget "http://downloads.sourceforge.net/project/moa-datastream/Datasets/Classification/
covtype Norm.arff.zip"
```

4.4.2. Command for Unzipping the Dataset

```
unzip covtypeNorm.arff.zip
```

4.5. Execute a Task of SAMOA on a Specific Platform

We have taken the PrequentialEvaluation task for training (building classifier) and testing (classifying) the stream data coming from covType dataset. The learning algorithm is the bagging algorithm already present in SAMOA library.

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

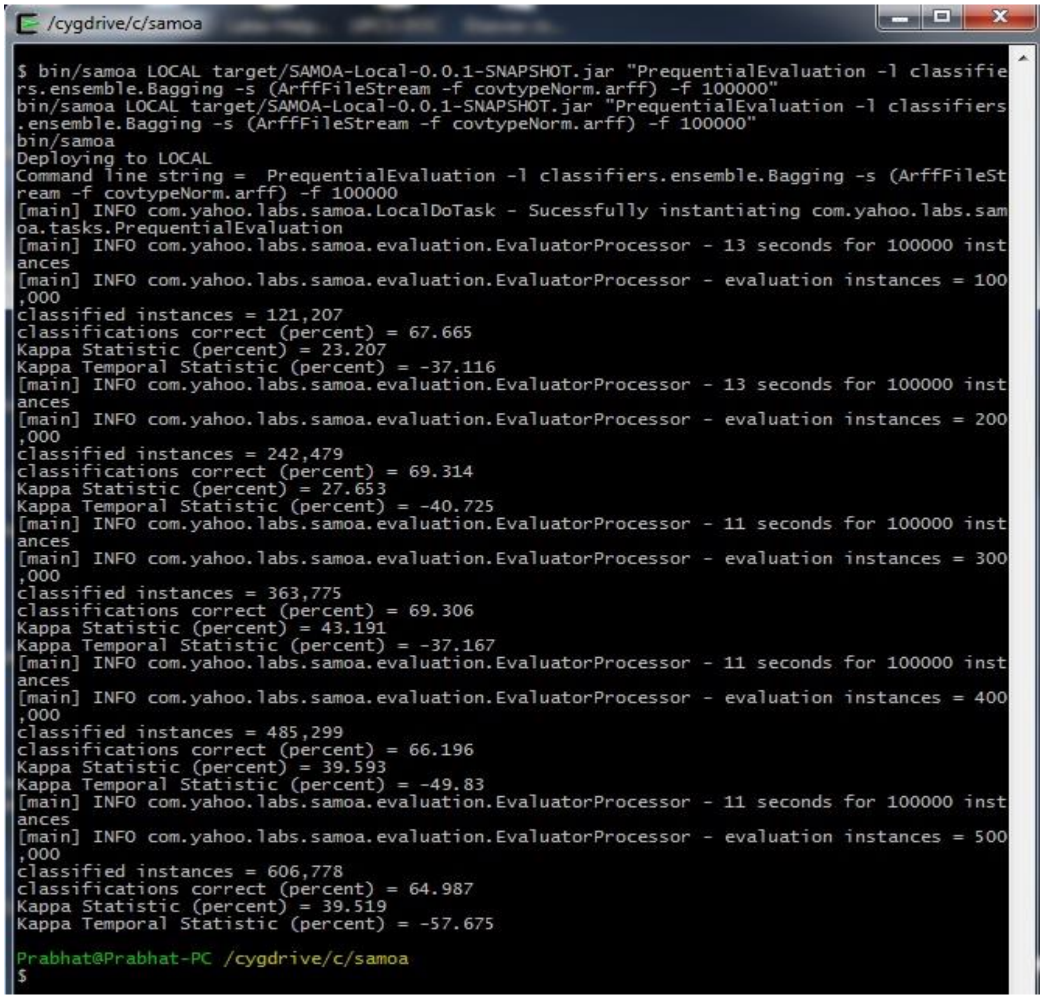
Here, *bin/samoa* is command to execute SAMOA script lying in bin directory of SAMOA installation, *'local'* specifies the deployment platform which is local in our use case, *target/SAMOA-Local-0.0.1-SNAPSHOT.jar* is the actual archival package containing the relevant class files for local deployment of SAMOA, *'PrequentialEvaluation*' is the name of task which is being execute. Also, *-l* represents the learner algorithm, *-s* represents the source of data stream. The value 100000 used with option *-f* represents the number of instances for each cycle of training. The evaluation results is plotted each 100,000 instances as shown in output snapshot in Figure 6.

5. Conclusion

This paper gives an introduction of stream analysis and machine learning, the requirement of distributed stream machine learning to tackle the challenge of velocity and scalability lying in Big Data. The focus of the paper is to give a small description of architecture and components of a recent and open source distributed framework named SAMOA which is used for Big Data stream mining and machine learning. The central goal of the paper is achieved in last by giving a detailed practical experience of installation and configuring SAMOA that ends with executing a sample PrequentialEvaluation task available in SAMOA.

6. Future Perspective

In this paper we have just executed one simple PrequentialEvaluation task available in SAMOA. This execution has been carried out on a single machine using local deployment. In future we would like to explore and experience the execution of other available learners (classifiers and clusters) and evaluators and their possible deployment on cluster of machine *i.e.*, on Storm or S4 platform. Also we are intending to exercise the available learning algorithms with other real time data and configuration to identify the comparison and advantages of techniques in SAMOA framework with that of other machine learning frameworks for stream analysis. As developer, also we would like to extend SAMOA library with new distributed machine learning algorithms and tasks as currently SAMOA contains too small set of learners and evaluator tasks.



```
$ bin/samoa LOCAL target/SAMOA-Local-0.0.1-SNAPSHOT.jar "PrequentialEvaluation -l classifie
rs.ensemble.Bagging -s (ArffFileStream -f covtypeNorm.arff) -f 100000"
bin/samoa LOCAL target/SAMOA-Local-0.0.1-SNAPSHOT.jar "PrequentialEvaluation -l classifiers
.ensemble.Bagging -s (ArffFileStream -f covtypeNorm.arff) -f 100000"
bin/samoa
Deploying to LOCAL
Command line string =  PrequentialEvaluation -l classifiers.ensemble.Bagging -s (ArffFileSt
ream -f covtypeNorm.arff) -f 100000
[main] INFO com.yahoo.labs.samoa.LocalDoTask - Sucessfully instantiating com.yahoo.labs.sam
oa.tasks.PrequentialEvaluation
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - 13 seconds for 100000 inst
ances
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - evaluation instances = 100
,000
classified instances = 121,207
classifications correct (percent) = 67.665
Kappa Statistic (percent) = 23.207
Kappa Temporal Statistic (percent) = -37.116
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - 13 seconds for 100000 inst
ances
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - evaluation instances = 200
,000
classified instances = 242,479
classifications correct (percent) = 69.314
Kappa Statistic (percent) = 27.653
Kappa Temporal Statistic (percent) = -40.725
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - 11 seconds for 100000 inst
ances
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - evaluation instances = 300
,000
classified instances = 363,775
classifications correct (percent) = 69.306
Kappa Statistic (percent) = 43.191
Kappa Temporal Statistic (percent) = -37.167
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - 11 seconds for 100000 inst
ances
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - evaluation instances = 400
,000
classified instances = 485,299
classifications correct (percent) = 66.196
Kappa Statistic (percent) = 39.593
Kappa Temporal Statistic (percent) = -49.83
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - 11 seconds for 100000 inst
ances
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - evaluation instances = 500
,000
classified instances = 606,778
classifications correct (percent) = 64.987
Kappa Statistic (percent) = 39.519
Kappa Temporal Statistic (percent) = -57.675

Prabhat@Prabhat-PC /cygdrive/c/samoa
$
```

Figure 6. Snapshot of Execution of PrequentialEvaluation Task on SAMOA

Acknowledgment

We thank to Dr. Arinto Murdopo and team to illustrate the concepts of SAMOA framework that enabled us to have a practical experience in executing the sample task of SAMOA.

References

[1] D. Tomar and S. Agarwal, A survey on Data Mining approaches for Healthcare, International Journal of Bio-Science & Bio-Technology. vol.5, no. 5, pp. 241-266 **(2013)**.

[2] S. Agarwal, Divya and G. N. Pandey, SVM based context awareness using body area sensor network for pervasive healthcare monitoring. Proceedings of the 1st International Conference on Intelligent Interactive Technologies and Multimedia, pp. 271-278 **(2010)**; New York, USA.

[3] D. Tomar and S. Agarwal, Predictive Model for diabetic patients using Hybrid Twin Support Vector Machine. Proceedings of the 5th International Conferences on Advances in Communication Network and Computing, **(2014)**. in press

[4] S. Khanna and S. Agarwal, An Integrated Approach towards the prediction of Likelihood of Diabetes. Proceedings of the International Conference on Machine Intelligence Research and Advancement, **(2013)**. in press.

[5] D. Tomar and S. Agarwal, Feature Selection based Least Square Twin Support Vector Machine for Diagnosis of Heart Disease. International Journal of Bio-Science & Bio-Technology. vol. 6, no. 2, pp. 69-82 **(2014)**.

[6] N. Rathore, D. Tomar and S. Agarwal, Predicting the survivability of breast cancer patients using ensemble approach. Proceedings of the IEEE International Conference on Issues and Challenges in Intelligent Computing Techniques, pp. 459-464 **(2014)** February; Ghaziabad, India.

[7] M. R. Wigan and R. Clarke, Big Data's Big Unintended Consequences. IEEE Computer Society. vol. 46, no. 6, pp. 46-53 **(2013)** June.

[8] J. McKendrick, Big Data, Big Challenges, Big Opportunities: 2012 IOUG Big Data Strategies Survey (White paper). Unisphere Research, a Division of Information Today, Inc., Unisphere Media Publishers, New Province, **(2012)** September;

[9] T. White, Hadoop: The Definitive Guide, O'Reilly Media Publishers, Yahoo Press, **(2012)** May.

[10] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen and T. Seidl, MOA: Massive Online Analysis, The Journal of Machine Learning. vol. 11, pp. 1601-1604 **(2010)**.

[11] Apache, What is Apache Mahout?. **(2014)**, April; http://mahout.apache.org. Accessed on July 2, 2014.

[12] T. Kraska, A. Talwalkar, J. Duchi, R. Griffith, M. J. Franklin and M. Jordan, MLBase: A Distributed Machine-Learning System. Proceedings of the Conference on Innovative Data Systems Research, **(2013)** January; California, USA.

[13] R. Kirkby, Improving Hoeffding Trees. PhD thesis, **(2007)** November; University of Waikato, New Zealand.

[14] MOA Massive Online Analysis. University of Waikato, **(2014)**, April; http://moa.cms.waikato.ac.nz. Accessed on July 5, 2014.

[15] Heywhoah, Vowpal Wabbit. **(2013)**; November; http://github.com/JohnLangford/vowpal_wabbit/wiki. Accessed on July 7, 2014.

[16] J. Jenkov, Maven Tutorial. **(2014)**; http://tutorials.jenkov.com/maven/maven-tutorial.html. Accessed on July 10, 2014.

[17] Wikipedia: the free encyclopedia, GitHub. **(2014)**; http://en.wikipedia.org/wiki/GitHub. Accessed on July 15, 2014.

[18] A. Murdopo, A. Severien, G. D. F. Morales and A. Bifet, SAMOA: Developer's Guide, Yahoo Labs, Barcelona, **(2013)** July.

[19] L. Neumeyer, B. Robbins, A. Nair and A. Kesari, S4: Distributed stream computing platform. Proceedings of the IEEE Conference on Data Mining Workshops (ICDMW), pp. 170–177 **(2010)** December; Sydney, NSW, Australia.

[20] Yahoo, Introducing SAMOA, an open source platform for mining big data streams. **(2014)**; http:// yahooeng.tumblr.com/post/65453012905/introducing-samoa-an-open-source-platform-for-mining. Accessed on July 19, 2014.

Authors

Bakshi Rohit Prasad, he is a research scholar in Information Technology Division of Indian Institute of Information Technology (IIIT), Allahabad, India His primary research interests are Data Mining, Machine Learning, Big Data Computing and Algorithms along with their applications in several domains.

Dr. Sonali Agarwal, Agarwal is working as an Assistant Professor in the Information Technology Division of Indian Institute of Information Technology (IIIT), Allahabad, India. Her primary research interests are in the areas of Data Mining, Data Warehousing, E Governance and Software Engineering. Her current focus in the last few years is on the research issues in Big Data Computing and its application.