SAMOA
Scalable Advanced Massive Online Analysis
Lingyun (Luke) Li
Big Data Streams

- High amount of data
- High speed of arrival
- Updated models at “real” time
- Potentially infinite sequence of data
- Change over time
Mining Big Data Streams

- Approximation algorithms:
  - Single pass
  - One data item at a time
  - Sub-linear space and time per data item
  - Small error with high probability

- Need a platform solution:
  - Distributed
  - Scalable
  - Support different algorithms & processing engines
SAMOA

- Scalable Advanced Massive Online Analysis
- Written in Java
- A platform for mining big data streams
  - Framework for developing new distributed stream mining algorithms
  - Framework for deploying algorithms on new distributed stream processing engines
SAMOA

- Library of state-of-the-art algorithms for distributed machine learning on streams
  - Classification - Vertical Hoeffding Tree (VHT), a distributed version of a streaming decision tree
  - Clustering - an algorithm based on CluStream
  - Regression - Adaptive Model Rules Regressor, a decision rule learner
  - Distributed sample-based frequent itemset mining
  - Meta-algorithms such as bagging and boosting
Taxonomy of data mining tools

- Machine Learning
  - Distributed
    - Batch
    - Stream
      - Hadoop
      - S4, Storm
      - Mahout
      - SAMOA
  - Non Distributed
    - Batch
      - R, WEKA, ...
    - Stream
      - MOA
SAMOA Architecture
Why Is SAMOA Interesting?

- Program once, run everywhere
  - Code and infrastructure reuse
- Model is always up to date
  - No system downtime
  - No complex backup/update process
  - No need to select update frequency
Developer API

```java
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);
```
Deployment

S4 bindings

SAMOA-S4.jar

samoa-s4-deployable.s4r

To S4 cluster

API. Algorithm developer depends only on this

SAMOA-API.jar

samoa-storm-deployable.jar

To Storm cluster

SAMOA-Storm.jar

Storm bindings
Download and Build SAMOA

```bash
$ git clone http://git.apache.org/incubator-samoa.git
$ cd incubator-samoa
$ mvn -Ps4 package
```
Download and Build SAMOA

```
$ git clone http://git.apache.org/incubator-samoa.git
$ cd incubator-samoa
$ mvn -Pstorm package
```
Download and Build SAMOA

```bash
~ $ git clone http://git.apache.org/incubator-samoa.git
~ $ cd incubator-samoa
~ $ mvn -Psamza package
```
Download and Build SAMOA

bin/samoa storm target/SAMOA-Storm-0.3.0-SNAPSHOT.jar
"PrequentialEvaluation"
-d /tmp/dump.csv
-i 10000000 -f 1000000
-l (classifiers.trees.VerticalHoeffdingTree -p 4 -k)
-s (generators.RandomTreeGenerator -r 1 -c 2 -o 10 -u 10)"

-\texttt{\textbf{-l}}: classifier to train
-\texttt{\textbf{-s}}: stream to learn from
-\texttt{\textbf{-e}}: classification performance evaluation method
-\texttt{\textbf{-i}}: maximum number of instances to test/train on (-1 = no limit)
-\texttt{\textbf{-f}}: number of instances between samples of the learning performance
-\texttt{\textbf{-n}}: evaluation name (default: PrequentialEvaluation_TimeStamp)
-\texttt{\textbf{-d}}: file to append intermediate csv results to
Case Study: Vertical Hoeffding Tree

- A distributed version of a streaming decision tree
- Uses the Hoeffding bound to decide the minimum number of arriving instances to achieve certain level of confidence in splitting the node
- Type of parallelism
  - Task
  - Data
    - Horizontal
    - Vertical

Sample email classifier decision tree
Task parallelism
Horizontal Parallelism
Vertical Parallelism

Stream → Model → Stats → Splits

Attributes → Stats → Stats → Stats
Advantages of Vertical Parallelism

- High level parallelism for High number of attributes (e.g., words)
- Reduced memory usage
  - Attribute counters are not replicated across several machines
- Parallelized split computation
Vertical Hoeffding Tree

Diagram showing components and connections within a vertical Hoeffding Tree, including source PI, model-aggregator PI, local-result, control, attribute, local-statistic PI, and result.
SAMOA Use Cases

- Data
  - Big fast data
  - Endless streams of data
  - Evolving data
- Updated models at real time
- Implement machine learning algorithms on different distributed stream processing engines
Summaries

- SAMOA: a platform for mining big data streams
- Supports the most common machine learning tasks
- Supports popular distributed stream processing engines (Storm, S4, Samza)
- Provides an API for implementing distributed streaming algorithms
- Available as an open-source Apache Project
Reference

- SAMOA: A Platform for Mining Big Data Streams, Gianmarco De Francisci Morales

Thank you!