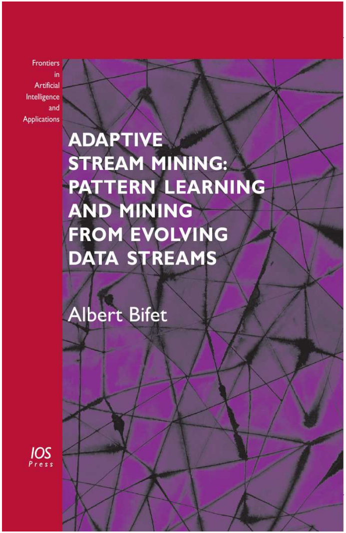
Mining Big Data in Real Time

Albert Bifet

Dortmund, 24 October 2013

Albert Bifet



2004-2009
Ph. D. Degree
UPC-Barcelona Tech
Advisors: Ricard Gavaldà and José L. Balcázar.

2009-2012
Post-Doctoral Researcher
University of Waikato, Hamilton, New Zealand

20011-2013
Researcher
Yahoo! Research Barcelona

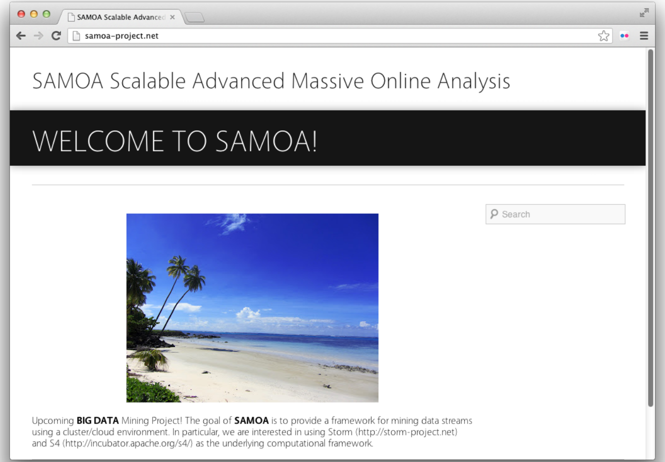
Albert Bifet

2004-2009
Ph. D. Degree
UPC-Barcelona Tech
Advisors: Ricard Gavaldà
and José L. Balcázar.

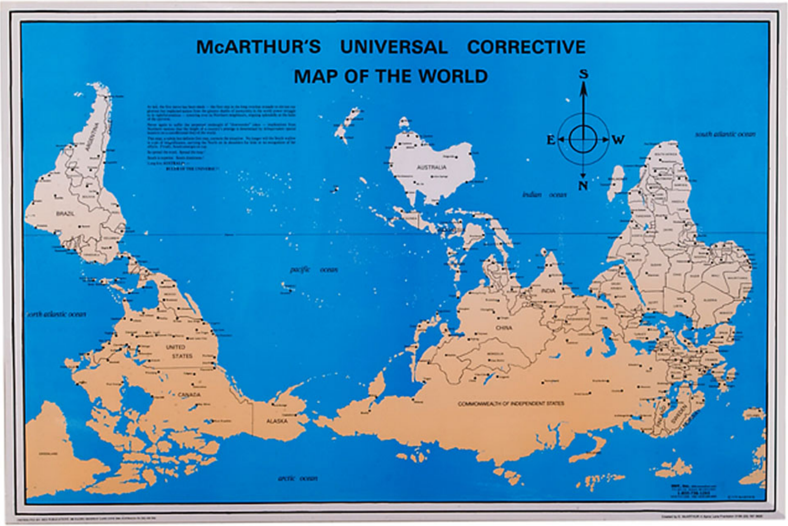
2009-2012
Post-Doctoral Researcher
University of Waikato,
Hamilton, New Zealand

20011-2013
Researcher
Yahoo! Research Barcelona

MOA-SAMOA


SAMOA Scalable Advanced
samoa-project.net
SAMOA Scalable Advanced Massive Online Analysis
WELCOME TO SAMOA!
Search
Upcoming BIG DATA Mining Project! The goal of SAMOA is to provide a framework for mining data streams using a cluster/cloud environment. In particular, we are interested in using Storm (http://storm-project.net) and S4 (http://incubator.apache.org/s4/) as the underlying computational framework.

New Zealand



McARTHUR'S UNIVERSAL CORRECTIVE
MAP OF THE WORLD
S
E
W
N
south atlantic ocean
indian ocean
pacific ocean
north atlantic ocean
arctic ocean
AUSTRALIA
BRAZIL
ARGENTINA
UNITED STATES
CANADA
ALASKA
CHINA
INDIA
COMMONWEALTH OF INDEPENDENT STATES
SWEDEN

Hamilton


Kaitaia
Whangarei
North Island
Auckland
Tauranga
Hamilton
Gisborne
New Plymouth
Napier
Hastings
Tasman Sea
Wanganui
Palmerston North
WELLINGTON
Nelson
Wesport
Blenheim
Greymouth
Kaikoura
Hokitika
Pacific
Ocean
South Island
Christchurch
Ashburton
Timaru
Oamaru
Gore
Dunedin
Invercargill
Stewart Island
300 km
©InterCarto

Outline

1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Outline

1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Big Data & Real Time

Big Data

McKinsey Global Institute (MGI) Report on Big Data, 2011.

Big data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze.

BIG Data

- Volume
- Variety
- Velocity

3 Vs

BIG Data

- Volume
- Variety
- Velocity
- Variability
- Value
- Veracity

6 Vs

Methodology

Distributed systems

Methodology

Paolo Boldi

Facebook Four degrees of separation

Big Data does not need big machines,
it needs big **intelligence**

Introduction: Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Introduction: Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Use a n-bit vector to memorize all the numbers ($O(n)$ space)

Introduction: Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Data Streams: $O(\log(n))$ space.

Introduction: Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Example

Puzzle: Finding Missing Numbers

- Let π be a permutation of $\{1, \ldots, n\}$.
- Let π_{-1} be π with one element missing.
- $\pi_{-1}[i]$ arrives in increasing order

Task: Determine the missing number

Data Streams: $O(\log(n))$ space. Store

$$\frac{n(n+1)}{2} - \sum_{j \leq i} \pi_{-1}[j].$$

Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Tools:

- approximation
- randomization, sampling
- sketching

Data Streams

Data Streams

- Sequence is potentially infinite
- High amount of data: sublinear space
- High speed of arrival: sublinear time per example
- Once an element from a data stream has been processed it is discarded or archived

Approximation algorithms

- Small error rate with high probability
- An algorithm $(\varepsilon, \delta)-$approximates F if it outputs $\tilde{F}$ for which $\Pr[|\tilde{F} - F| > \varepsilon F] < \delta$.

Data Streams Approximation Algorithms

1011000111 $\boxed{1010101}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon} \log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Streams Approximation Algorithms

10110001111 $\boxed{0101011}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Streams Approximation Algorithms

101100011110 $\boxed{1010111}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Streams Approximation Algorithms

1011000111101 $\boxed{0101110}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Streams Approximation Algorithms

10110001111010 $\boxed{1011101}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

Data Streams Approximation Algorithms

1011000111 10101 $\boxed{0111010}$

Sliding Window

We can maintain simple statistics over sliding windows, using $O(\frac{1}{\varepsilon}\log^2 N)$ space, where

- N is the length of the sliding window
- ε is the accuracy parameter

M. Datar, A. Gionis, P. Indyk, and R. Motwani.
Maintaining stream statistics over sliding windows. 2002

What is MOA?

{M}assive {O}nline {A}nalysis is a framework for online learning from data streams.

- It is closely related to WEKA
- It includes a collection of offline and online as well as tools for evaluation:
  - classification
  - clustering
- Easy to extend
- Easy to design and run experiments

History - timeline

WEKA

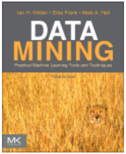
- 1993 - WEKA : project starts (Ian Witten)
- 1996 - First public release of WEKA in C
- Early 1997 - decision was made to rewrite WEKA in Java
- Mid 1999 - WEKA 3 (100% Java) released

MOA

- Nov 2007 - First public release of MOA: Richard Kirkby Thesis
- 2009 - MOA Concept Drift
- 2010 - MOA Clustering
- 2011 - MOA Graph Mining, Multi-label classification, Twitter Reader, Active Learning
- 2013 - MOA Outlier

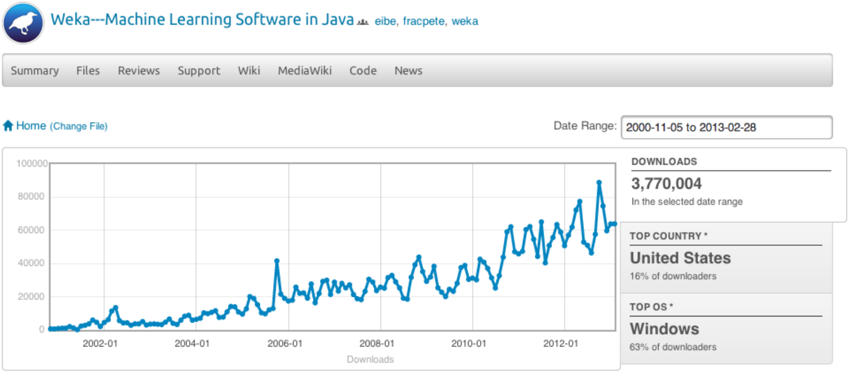
WEKA

- Waikato Environment for Knowledge Analysis
- Collection of state-of-the-art machine learning algorithms and data processing tools implemented in Java
  - Released under the GPL
- Support for the whole process of experimental data mining
  - Preparation of input data
  - Statistical evaluation of learning schemes
  - Visualization of input data and the result of learning



- Used for education, research and applications
- Complements “Data Mining” by Witten & Frank & Hall

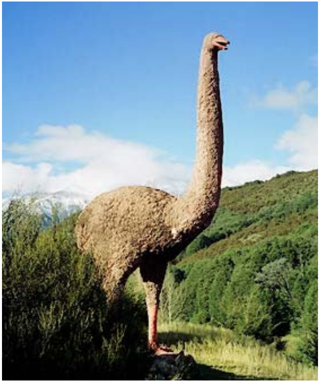
WEKA: Impact Downloads



WEKA: the bird

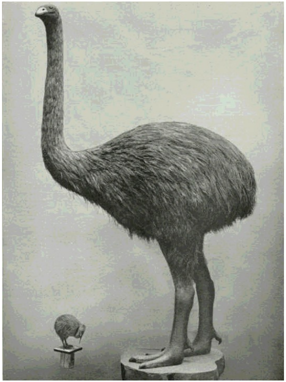
MOA: the bird

The Moa (another native NZ bird) is not only flightless, like the Weka, but also extinct.



MOA: the bird

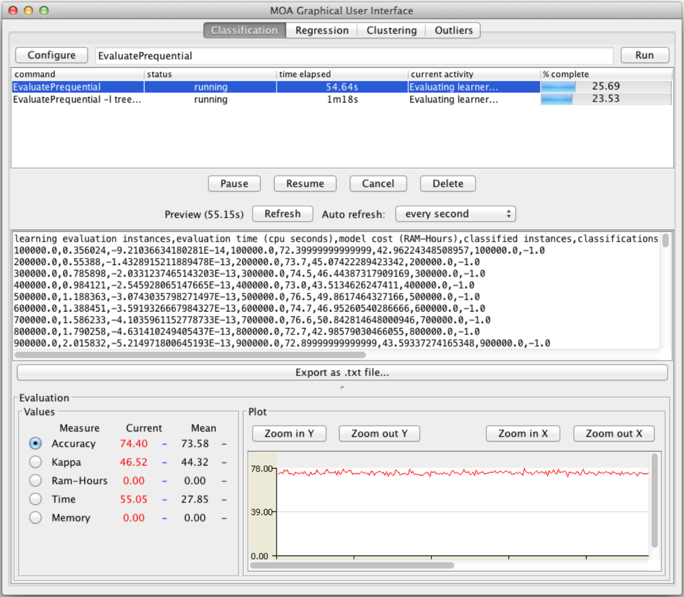
The Moa (another native NZ bird) is not only flightless, like the Weka, but also extinct.



MOA: the bird

The Moa (another native NZ bird) is not only flightless, like the Weka, but also extinct.

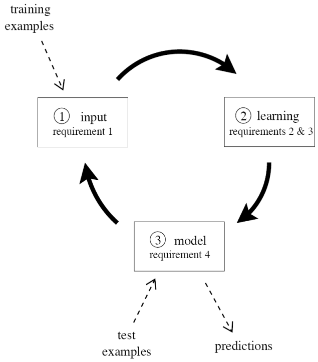
Classification Experimental Setting



Classification Experimental Setting

Evaluation procedures for Data Streams

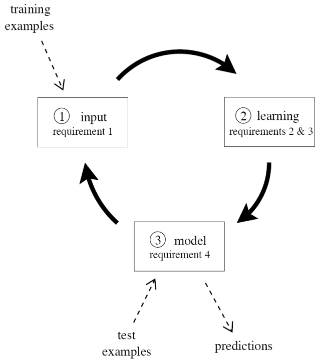
- Holdout
- Interleaved Test-Then-Train or Prequential



Classification Experimental Setting

Data Sources

- Random Tree Generator
- Random RBF Generator
- LED Generator
- Waveform Generator
- Hyperplane
- SEA Generator
- STAGGER Generator



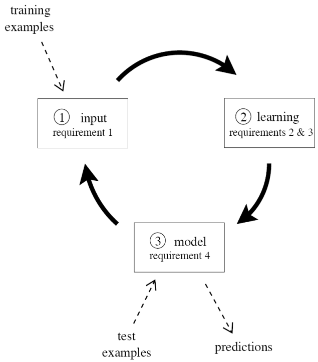
Classification Experimental Setting

Classifiers

- Naive Bayes
- Decision stumps
- Hoeffding Tree
- Hoeffding Option Tree
- Bagging and Boosting
- ADWIN Bagging and Leveraging Bagging

Prediction strategies

- Majority class
- Naive Bayes Leaves
- Adaptive Hybrid



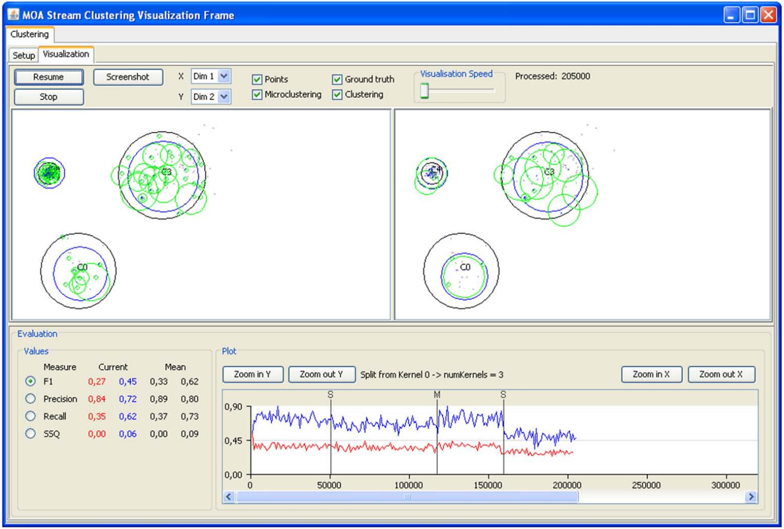
RAM-Hours

RAM-Hour

Every GB of RAM deployed for 1 hour

Cloud Computing Rental Cost Options

Clustering Experimental Setting



Clustering Experimental Setting

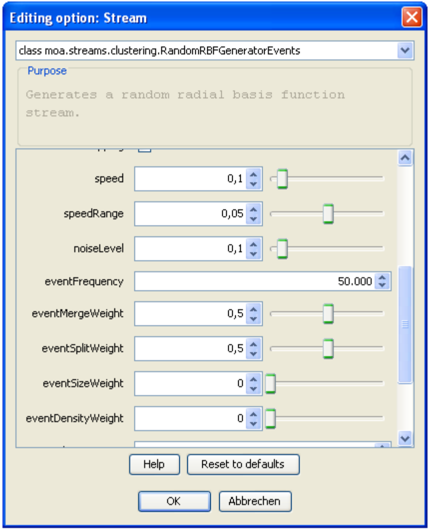
Internal measures	External measures
Gamma	Rand statistic
C Index	Jaccard coefficient
Point-Biserial	Folkes and Mallow Index
Log Likelihood	Hubert Γ statistics
Dunn's Index	Minkowski score
Tau	Purity
Tau A	van Dongen criterion
Tau C	V-measure
Somer's Gamma	Completeness
Ratio of Repetition	Homogeneity
Modified Ratio of Repetition	Variation of information
Adjusted Ratio of Clustering	Mutual information
Fagan's Index	Class-based entropy
Deviation Index	Cluster-based entropy
Z-Score Index	Precision
D Index	Recall
Silhouette coefficient	F-measure

Table : Internal and external clustering evaluation measures.

Clustering Experimental Setting

Clusterers

- StreamKM++
- CluStream
- ClusTree
- Den-Stream
- D-Stream
- CobWeb



Web

http://www.moa.cms.waikato.ac.nz

Easy Design of a MOA classifier

- `void resetLearningImpl ()`
- `void trainOnInstanceImpl (Instance inst)`
- `double[] getVotesForInstance (Instance i)`

Easy Design of a MOA clusterer

- `void resetLearningImpl ()`
- `void trainOnInstanceImpl (Instance inst)`
- `Clustering getClusteringResult()`

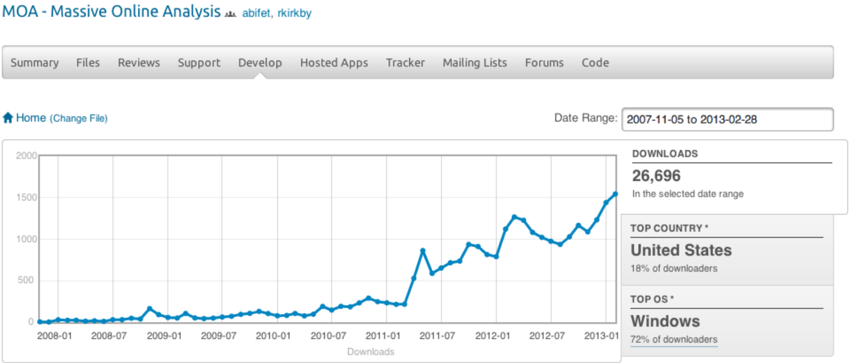
Extensions of MOA

- Multi-label Classification
- Active Learning
- Regression
- Closed Frequent Graph Mining
- Twitter Sentiment Analysis

Challenges for bigger data streams

Sampling and distributed systems (Map-Reduce, Hadoop, S4)

MOA: Impact Downloads



Outline

1. MOA: Massive Online Analysis

2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining

3. Summary and Future Work: SAMOA

Outline

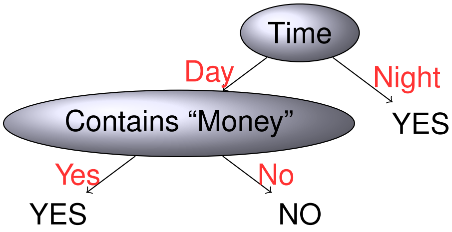
1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Hoeffding Trees

Hoeffding Tree : VFDT

Pedro Domingos and Geoff Hulten.
Mining high-speed data streams. 2000

- With high probability, constructs an identical model that a traditional (greedy) method would learn
- With theoretical guarantees on the error rate



Hoeffding Naive Bayes Tree

Hoeffding Tree

Majority Class learner at leaves

Hoeffding Naive Bayes Tree

G. Holmes, R. Kirkby, and B. Pfahringer.
Stress-testing Hoeffding trees, 2005.

- monitors accuracy of a Majority Class learner
- monitors accuracy of a Naive Bayes learner
- predicts using the most accurate method

Bagging

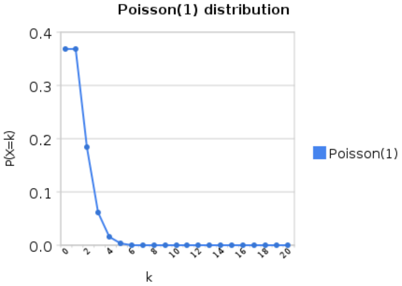


Figure : Poisson(1) Distribution.

Bagging builds a set of M base models, with a bootstrap sample created by drawing random samples with replacement.

Bagging

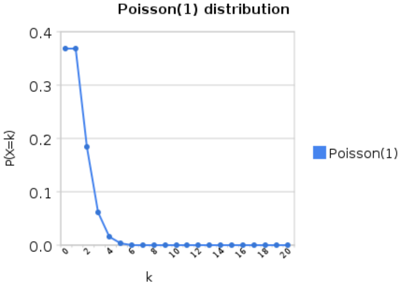


Figure : Poisson(1) Distribution.

Each base model's training set contains each of the original training example K times where $P(K = k)$ follows a binomial distribution.

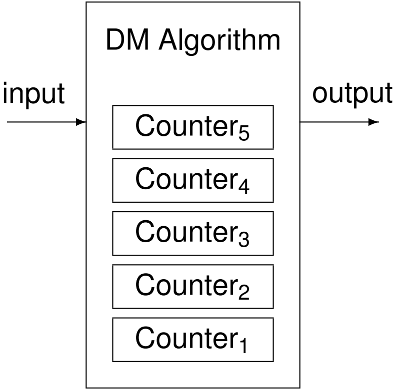
Oza and Russell's *Online Bagging* for M models

Initialize base models h_m for all $m \in \{1, 2, ..., M\}$
for all training examples **do**
  for $m = 1, 2, ..., M$ **do**
    Set $w =$ *Poisson*(1)
    Update h_m with the current example with weight w

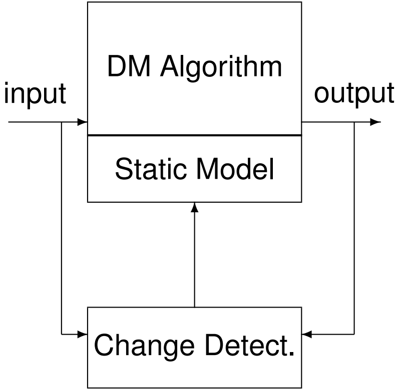
anytime output:
return hypothesis: $h_{fin}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} I(h_t(x) = y)$

Data Mining Algorithms with Concept Drift

No Concept Drift

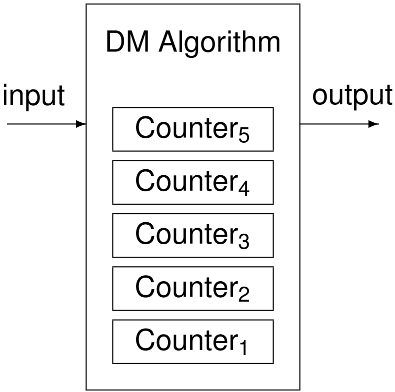


Concept Drift

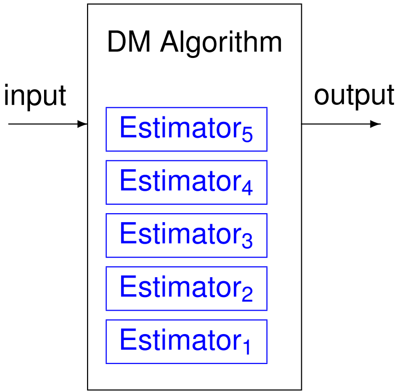


Data Mining Algorithms with Concept Drift

No Concept Drift



Concept Drift



Optimal Change Detector and Predictor

- High accuracy
- Low false positives and false negatives ratios
- Theoretical guarantees

- Fast detection of change
- Low computational cost: minimum space and time needed

- No parameters needed

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
      do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
            repeat Drop elements from the tail of W
               until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
                  for every split of W into W = W_0 · W_1
            Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1}$ $W_1 = \boxed{01010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
      do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
         repeat Drop elements from the tail of W
           until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
              for every split of W into W = W_0 · W_1
         Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{10}$ $W_1 = \boxed{1010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

Initialize Window W
for each $t > 0$
  do $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
    repeat Drop elements from the tail of W
      until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$ holds
        for every split of W into $W = W_0 \cdot W_1$
    Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{101}$ $W_1 = \boxed{010110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

Initialize Window W
for each $t > 0$
  do $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
    repeat Drop elements from the tail of W
      until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$ holds
        for every split of W into $W = W_0 \cdot W_1$
    Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

Example

W= $\boxed{101010110111111}$

W_0= $\boxed{1010}$ $W_1 = \boxed{10110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
    do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
        repeat Drop elements from the tail of W
          until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
            for every split of W into W = W_0 · W_1
        Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{101010}$ $W_1 = \boxed{110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
    do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
        repeat Drop elements from the tail of W
          until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
            for every split of W into W = W_0 · W_1
        Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W =$ $\boxed{101010110111111}$

$W_0 =$ $\boxed{101010}$ $W_1 = \boxed{110111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

Initialize Window W
for each $t > 0$
  do $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
    repeat Drop elements from the tail of W
      until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$ holds
        for every split of W into $W = W_0 \cdot W_1$
    Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{1010101}$ $W_1 = \boxed{10111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

Initialize Window W
for each $t > 0$
 do $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
  repeat Drop elements from the tail of W
   until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$ holds
    for every split of W into $W = W_0 \cdot W_1$
  Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$

$W_0 = \boxed{10101011}$ $W_1 = \boxed{0111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
    do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
        repeat Drop elements from the tail of W
          until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
            for every split of W into W = W_0 · W_1
        Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{101010110111111}$ $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$: CHANGE DET.!

$W_0 = \boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

Initialize Window W
for each $t > 0$
  do $W \leftarrow W \cup \{x_t\}$ (i.e., add x_t to the head of W)
    repeat Drop elements from the tail of W
      until $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \varepsilon_c$ holds
        for every split of W into $W = W_0 \cdot W_1$
    Output $\hat{\mu}_W$

Algorithm ADaptive Sliding WINdow

Example

W= $\boxed{1010101101111111}$ Drop elements from the tail of W

W_0= $\boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
      do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
         repeat Drop elements from the tail of W
           until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
             for every split of W into W = W_0 · W_1
         Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Example

$W = \boxed{01010110111111}$ Drop elements from the tail of W

$W_0 = \boxed{101010110}$ $W_1 = \boxed{111111}$

ADWIN: ADAPTIVE WINDOWING ALGORITHM

```
Initialize Window W
for each t > 0
    do W ← W ∪ {x_t} (i.e., add x_t to the head of W)
        repeat Drop elements from the tail of W
          until |μ̂_{W_0} − μ̂_{W_1}| ≥ ε_c holds
            for every split of W into W = W_0 · W_1
        Output μ̂_W
```

Algorithm ADaptive Sliding WINdow

Theorem

At every time step we have:

1. (False positive rate bound). *If μ_t remains constant within W, the probability that* ADWIN *shrinks the window at this step is at most δ.*
2. (False negative rate bound). *Suppose that for* some *partition of W in two parts $W_0 W_1$ (where W_1 contains the most recent items) we have $|\mu_{W_0} - \mu_{W_1}| > 2\varepsilon_c$. Then with probability* $1 - \delta$ ADWIN *shrinks W to W_1, or shorter.*

ADWIN tunes itself to the data stream at hand, with no need for the user to hardwire or precompute parameters.

Algorithm ADaptive Sliding WINdow

ADWIN using a Data Stream Sliding Window Model,

- can provide the exact counts of 1's in $O(1)$ time per point.
- tries $O(\log W)$ cutpoints
- uses $O(\frac{1}{\varepsilon} \log W)$ memory words
- the processing time per example is $O(\log W)$ (amortized and worst-case).

Sliding Window Model

	1010101	101	11	1	1
Content:	4	2	2	1	1
Capacity:	7	3	2	1	1

Decision Trees: Hoeffding Adaptive Tree

Hoeffding Adaptive Tree:

- replace frequency statistics counters by estimators
  - don't need a window to store examples, due to the fact that we maintain the statistics data needed with estimators
- change the way of checking the substitution of alternate subtrees, using a change detector with theoretical guarantees

Advantages over CVFDT:

1. Theoretical guarantees
2. No Parameters

ADWIN Bagging (KDD'09)

ADWIN

An adaptive sliding window whose size is recomputed online according to the rate of change observed.

ADWIN has rigorous guarantees (theorems)

- On ratio of false positives and negatives
- On the relation of the size of the current window and change rates

ADWIN Bagging

When a change is detected, the worst classifier is removed and a new classifier is added.

Leveraging Bagging for Evolving Data Streams

Randomization as a powerful tool to increase accuracy and diversity

There are three ways of using randomization:

- Manipulating the input data
- Manipulating the classifier algorithms
- Manipulating the output targets

Leveraging Bagging for Evolving Data Streams

Leveraging Bagging

- Using *Poisson*(λ)

Leveraging Bagging MC

- Using *Poisson*(λ) and Random Output Codes

Fast Leveraging Bagging ME

- if an instance is misclassified: weight = 1
- if not: weight = $e_T/(1-e_T)$,

Empirical evaluation

	Accuracy	RAM-Hours
Hoeffding Tree	74.03%	0.01
Online Bagging	77.15%	2.98
`ADWIN` Bagging	79.24%	1.48
Leveraging Bagging	85.54%	20.17
Leveraging Bagging MC	85.37%	22.04
Leveraging Bagging ME	80.77%	0.87

Leveraging Bagging

- Leveraging Bagging
  - Using *Poisson*(λ)
- Leveraging Bagging MC
  - Using *Poisson*(λ) and Random Output Codes
- Leveraging Bagging ME
  - Using weight 1 if misclassified, otherwise $e_T/(1-e_T)$

Outline

1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Active Learning

I. Zliobaitė, A. Bifet, B. Pfahringer, G. Holmes
Active learning with evolving streaming data

ACTIVE LEARNING FRAMEWORK

Input: labeling budget B and strategy parameters

for each X_t - incoming instance,
  do if ACTIVE LEARNING STRATEGY$(X_t, B, \ldots)$ = **true**
    then request the true label y_t of instance X_t
      train classifier L with (X_t, y_t)
      if L_n exists **then** train classifier L_n with (X_t, y_t)
      if change warning is signaled
        then start a new classifier L_n
      if change is detected
        then replace classifier L with L_n

Active Learning

I. Žliobaitė, A. Bifet, B. Pfahringer, G. Holmes
Active learning with evolving streaming data

	Controlling Budget	Instance space Coverage
Random	present	full
Fixed uncertainty	no	fragment
Variable uncertainty	handled	fragment
Randomized uncertainty	handled	full

Table : Summary of strategies.

Outline

1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Multi-label classification

- Binary Classification: e.g. is this a beach? $\in$ {`No`, `Yes`}
- Multi-class Classification: e.g. what is this? $\in$ {`Beach`, `Forest`, `City`, `People`}
- Multi-label Classification: e.g. which of these? $\subseteq$ {`Beach`, `Forest`, `City`, `People`}

Methods for Multi-label Classification

Problem Transformation: Using off-the-shelf binary / multi-class classifiers for multi-label learning.

- **Binary Relevance method (BR)**
  - One binary classifier for each label:
    - simple; flexible; fast but does not explicitly model label dependencies
- **Label Powerset method (LP)**
  - One multi-class classifier; one class for each labelset

Data Streams Multi-label Classification

- **Adaptive Ensembles of Classifier Chains (ECC)**
  - Hoeffding trees as base-classifiers reset classifiers based on current performance / concept drift
- **Multi-label Hoeffding Tree**
  - Label Powerset method (LP) at the leaves an ensemble strategy to deal with concept drift
- **MOA Multi-label Setting**
  - generating synthetic multi-label data streams
  - setting a benchmark on real-world and synthetic data

Outline

1. MOA: Massive Online Analysis
2. Adaptive Size Sliding Window Learning
   - Classification
   - Active Learning
   - Multi-label Classification
   - Frequent Pattern Mining
3. Summary and Future Work: SAMOA

Pattern Mining

Dataset Example

Document	Patterns	Class
d1	abce	yes
d2	cde	no
d3	abce	yes
d4	acde	no
d5	abcde	no
d6	bcd	yes

Classification using patterns: mapping from patterns to vectors of attributes

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	**Frequent**
6	c
5	e,ce
4	a,ac,ae,ace
4	b,bc
4	d,cd
3	ab,abc,abe be,bce,abce
3	de,cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent	Gen	Closed
6	c	c	c
5	e,ce	e	ce
4	a,ac,ae,ace	a	ace
4	b,bc	b	bc
4	d,cd	d	cd
3	ab,abc,abe be,bce,abce	ab be	abce
3	de,cde	de	cde

Itemset Mining

d1	abce
d2	cde
d3	abce
d4	acde
d5	abcde
d6	bcd

Support	Frequent	Gen	Closed	Max
6	c	c	c	
5	e,ce	e	ce	
4	a,ac,ae,ace	a	ace	
4	b,bc	b	bc	
4	d,cd	d	cd	
3	ab,abc,abe	ab		
	be,bce,abce	be	abce	abce
3	de,cde	de	cde	cde

Graph Coresets KDD'11

Coreset of a set P with respect to some problem

Small subset that approximates the original set P.

- Solving the problem for the coreset provides an approximate solution for the problem on P.

δ-tolerance Closed Graph

A graph g is *δ-tolerance closed* if none of its proper frequent supergraphs has a weighted support $\leq (1-\delta) \cdot support(g)$.

- Maximal graph: 1-tolerance closed graph
- Closed graph: 0-tolerance closed graph.

Graph Coresets KDD'11

Relative support of a closed graph

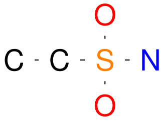
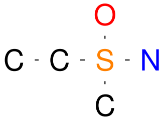
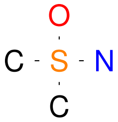
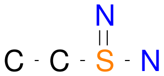
Support of a graph minus the relative support of its closed supergraphs.

- The sum of the closed supergraphs' relative supports of a graph is equal to its own support.

(s, δ)-coreset for the problem of computing closed graphs

Weighted multiset of frequent δ-tolerance closed graphs with minimum support s using their relative support as a weight.

Graph Dataset

Transaction Id	Graph	Weight
1		1
2		1
3		1
4		1

Graph Coresets

Graph	Relative Support	Support
C - C - S - N	3	3
O C - S - N	3	3
N ‖ C - S	3	3

Table : Example of a coreset with minimum support 50% and $\delta = 1$

Graph Coresets

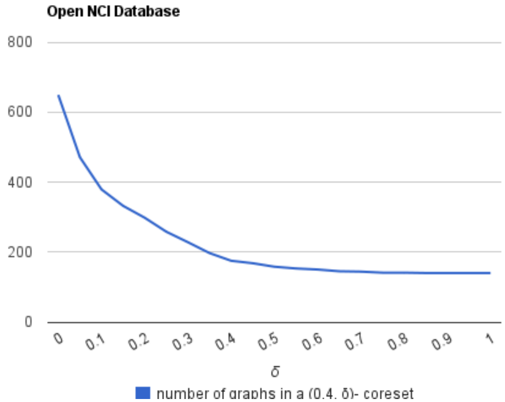
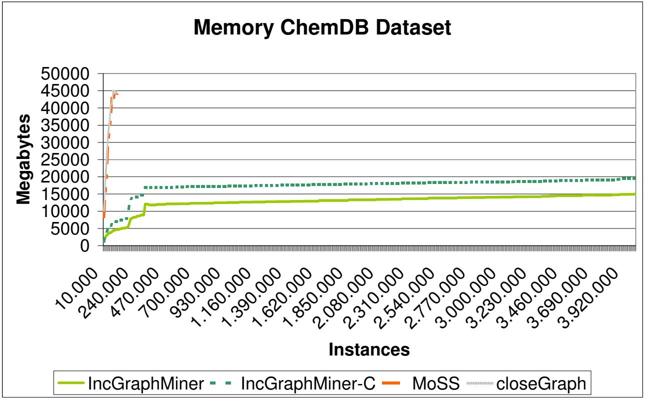
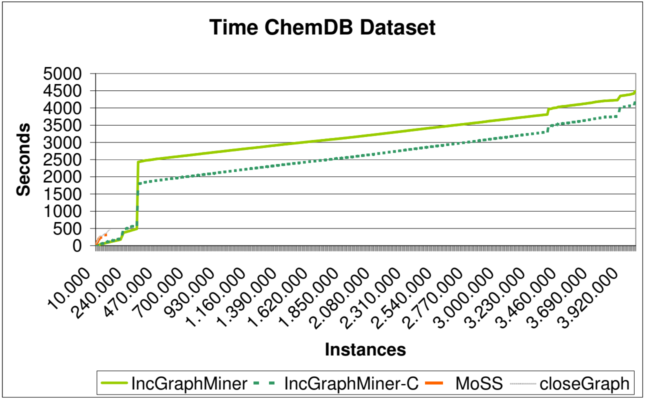


Figure : Number of graphs in $(40\%, \delta)$ for NCI.

ChemDB dataset



ChemDB dataset



Outline

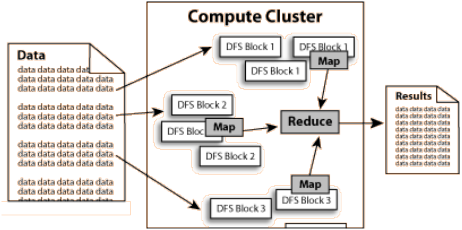
New Techniques: Distributed Systems

Hadoop, S4 and Storm

Hadoop

Hadoop

Hadoop



Hadoop architecture

Apache Mahout

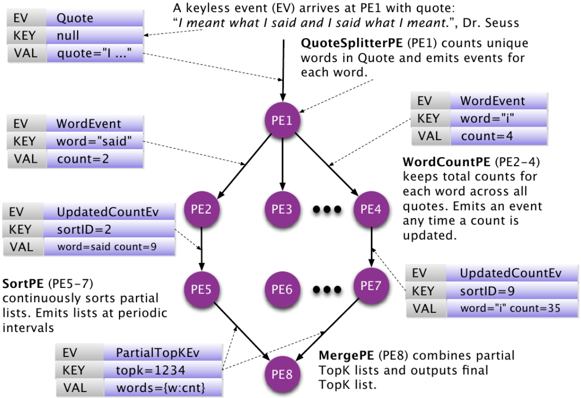
Mahout: open source framework

Apache S4

S4 distributed stream computing platform

Apache S4

Apache S4

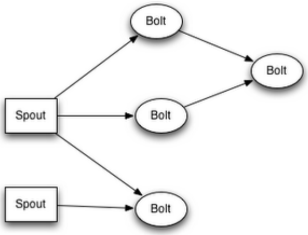


PE ID	PE Name	Key Tuple
PE1	QuoteSplitterPE	null
PE2	WordCountPE	word="said"
PE4	WordCountPE	word="i"
PE5	SortPE	sortID=2
PE7	SortPE	sortID=9
PE8	MergePE	topK=1234

Storm

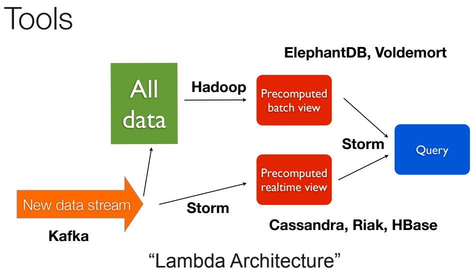
Storm from Twitter

Storm



Stream, Spout, Bolt, Topology

Storm

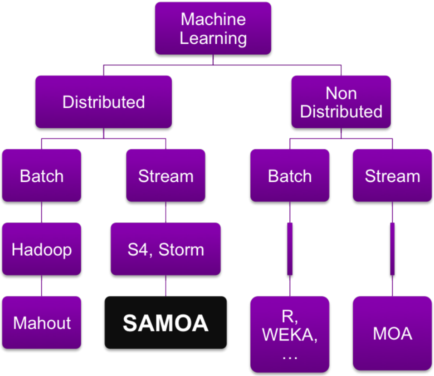


Runaway complexity in Big Data
Nathan Marz, 2012

SAMOA

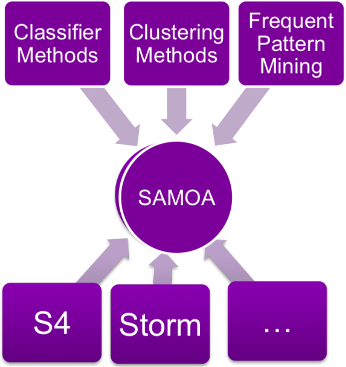
SAMOA: MOA + S4/Storm

SAMOA

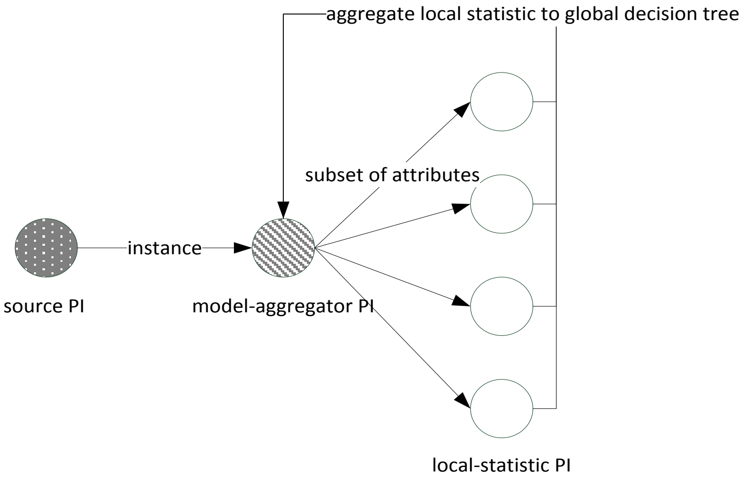


Machine Learning
Distributed
Non Distributed
Batch
Stream
Batch
Stream
Hadoop
S4, Storm
Mahout
SAMOA
R, WEKA, ...
MOA

SAMOA


Classifier Methods
Clustering Methods
Frequent Pattern Mining
SAMOA
S4
Storm
...

Vertical Hoeffding Tree



Summary

{M}assive {O}nline {A}nalysis is a framework for online learning from data streams.

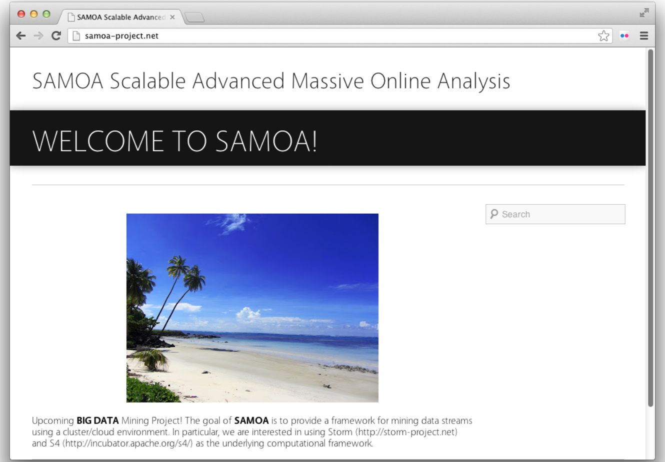
http://moa.cs.waikato.ac.nz

- It is closely related to WEKA
- It includes a collection of offline and online as well as tools for evaluation:
  - classification
  - clustering
  - frequent pattern mining
- MOA deals with evolving data streams
- MOA is easy to use and extend

SAMOA

SAMOA: MOA + S4/Storm

Thanks!



SAMOA Scalable Advanced
samoa-project.net
SAMOA Scalable Advanced Massive Online Analysis
WELCOME TO SAMOA!
Search
Upcoming BIG DATA Mining Project! The goal of SAMOA is to provide a framework for mining data streams using a cluster/cloud environment. In particular, we are interested in using Storm (http://storm-project.net) and S4 (http://incubator.apache.org/s4/) as the underlying computational framework.