Massive Online Analytics (MOA) for the Internet of Things (IoT)

Albert Bifet (@abifet)

Telecom ParisTech, 7 September 2017
IoT Setting
IoT: sensors and actuators connected by networks to computing systems.

- Gartner predicts 20.8 billion IoT devices by 2020.
- IDC projects 32 billion IoT devices by 2020
IoT Applications For Energy Management
IoT Applications For Connected/Smart Home
IoT Applications For Smart Cities
IoT Applications
For Industrial Automation
IOT AND INDUSTRY 4.0

- Interoperability: IoT
- Information transparency: virtual copy of the physical world
- Technical assistance: support human decisions
- Decentralized decisions: make decisions on their own
IoT versus Big Data

Google Trends
Comparer

Internet des objets
Domaine d'étude

Big data
Sujet

Ajouter une comparaison

Dans tous les pays
Cinq dernières années
Toutes les catégories
Recherche sur le Web

Évolution de l'intérêt pour cette recherche
Moyennes
26 févr. 2013
17 nov. 2013
9 août 2015
Data Set

Classifier Algorithm builds Model

Model

Analytic Standard Approach

Finite training sets
Static models
Data Stream Approach

Infinite training sets
Dynamic models
Pain Points

• Need to **retrain**!
  • Things change over time
  • How often?
• Data unused until next update!
  • Value of data wasted
IoT Stream Mining

- Maintain models online
  - Incorporate data on the fly
  - Unbounded training sets
  - Resource efficient
  - Detect changes and adapts
  - Dynamic models
Approximation Algorithms

- General idea, good for streaming algorithms
- Small error $\varepsilon$ with high probability $1-\delta$
  - True hypothesis $H$, and learned hypothesis $\hat{H}$
  - $\Pr[ |H - \hat{H}| < \varepsilon|H| ] > 1-\delta$
Approximation Algorithms

• What is the largest number that we can store in 8 bits?

\[ \begin{array}{cccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{array} \]
What is the largest number that we can store in 8 bits?

It is possible to use a small counter to keep approximate counts of large numbers. The resulting expected error can be rather precisely controlled. An example is given in which 8-bit counters (bytes) are used to keep track of as many as 130,000 events with a relative error which is substantially independent of the number $n$ of events. This relative error can be expected to be 24 percent or less 95 percent of the time (i.e. $\sigma = n/8$). The techniques could be used to advantage in multichannel counting hardware or software used for the monitoring of experiments or processes.
Approximation Algorithms

- What is the largest number that we can store in 8 bits?

\[ f(x) = \frac{\log(1 + x)}{\log(2)} \]

\( f(0) = 0, f(1) = 1 \)
Approximation Algorithms

- What is the largest number that we can store in 8 bits?

\[ f(x) = \log\left(1 + \frac{x}{30}\right)/\log\left(1 + \frac{1}{30}\right) \]

\[ f(0) = 0, f(1) = 1 \]
Approximation Algorithms

- What is the largest number that we can store in 8 bits?

\[ f(x) = \frac{\log(1 + x/30)}{\log(1 + 1/30)} \]

\[ f(0) = 0, f(1) = 1 \]
Approximation Algorithms

**MORRIS APPROXIMATE COUNTING ALGORITHM**

1. Init counter $c \leftarrow 0$
2. for every event in the stream
3. do $\text{rand} =$ random number between 0 and 1
4. if $\text{rand} < p$
5. then $c \leftarrow c + 1$

- What is the largest number that we can store in 8 bits?
Approximation Algorithms

101100011110101 0111010

**Sliding Window**

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

- $N$ is the length of the sliding window
- $\epsilon$ is the accuracy parameter

---

Maintaining stream statistics over sliding windows. 2002
WHAT IS MOA?
MOA

- Massive Online Analysis 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, regression
  - clustering, frequent pattern mining

- Easy to extend, design and run experiments.
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.
STREAM SETTING

• Process an example at a time, and inspect it only once (at most)
• Use a limited amount of memory
• Work in a limited amount of time
• Be ready to predict at any point
STREAM EVALUATION

• Holdout Evaluation

• Interleaved Test-Then-Train or Prequential
STREAM EVALUATION

Holdout an independent test set

• Apply the current decision model to the test set, at regular time intervals

• The loss estimated in the holdout is an unbiased estimator
STREAM EVALUATION

Prequential Evaluation

- The error of a model is computed from the sequence of examples.
- For each example in the stream, the actual model makes a prediction based only on the example attribute-values.

\[ S = \sum_{i=1}^{n} L(y_i, \hat{y}_i). \]
COMMAND LINE


- This command creates a comma separated values file:
  
  - training the DecisionStump classifier on the WaveformGenerator data,
  - using the first 100 thousand examples for testing,
  - training on a total of 100 million examples,
  - and testing every one million examples
Classification
Definition

Given a set of training examples belonging to \( n \) different classes, a classifier algorithm builds a model that predicts for every unlabeled instance \( x \) the class \( C \) to which it belongs.

Examples

• Email spam filter
• Twitter sentiment analyzer
Naïve Bayes

• Based on Bayes’ theorem

\[ P(C|x) = \frac{P(x|C)P(C)}{P(x)} \]

posterior = \text{likelihood} \times \text{prior} / \text{evidence}

• Probability of observing feature \( x_i \) given class \( C \)

• Prior class probability \( P(C) \)

• Just counting!

\[ P(C|x) \propto \prod_{x_i \in x} P(x_i|C)P(C) \]

\[ C = \arg \max_C P(C|x) \]
Perceptron

- Linear classifier
- Data stream: \( \langle \vec{x}_i, y_i \rangle \)
- \( \tilde{y}_i = h_{\vec{w}}(\vec{x}_i) = \sigma(\vec{w}^T \vec{x}_i) \)
- \( \sigma(x) = 1/(1+e^{-x}) \quad \sigma' = \sigma(x)(1-\sigma(x)) \)
- Minimize MSE \( J(\vec{w}) = \frac{1}{2} \sum (y_i - \tilde{y}_i)^2 \)
- SGD \( \vec{w}_{i+1} = \vec{w}_i - \eta \nabla J \vec{x}_i \)
  - \( \nabla J = -(y_i - \tilde{y}_i)\tilde{y}_i(1-\tilde{y}_i) \)
  - \( \vec{w}_{i+1} = \vec{w}_i + \eta (y_i - \tilde{y}_i)\tilde{y}_i(1-\tilde{y}_i)\vec{x}_i \)
Decision Tree

- Each node tests a feature
- Each branch represents a value
- Each leaf assigns a class
- Greedy recursive induction
  - Sort all examples through tree
  - \( x_i \) = most discriminative attribute
  - New node for \( x_i \), new branch for each value, leaf assigns majority class
  - Stop if no error | limit on #instances

Car deal?
HOEFFDING TREE

- Sample of stream enough for near optimal decision
- Estimate merit of alternatives from prefix of stream
- Choose sample size based on statistical principles
- When to expand a leaf?
  - Let $x_1$ be the most informative attribute, $x_2$ the second most informative one
  - Hoeffding bound: split if $G(x_1) - G(x_2) > \varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$
Regression
Definition

Given a set of training examples with a numeric label, a regression algorithm builds a model that predicts for every unlabeled instance $x$ the value with high accuracy $y=f(x)$.

Examples
- Stock price
- Airplane delay
Perceptron

- Linear regressor
- Data stream: $\langle \tilde{x}_i, y_i \rangle$
- $\tilde{y}_i = h_{\tilde{w}}(\tilde{x}_i) = \tilde{w}^T \tilde{x}_i$
- Minimize MSE $J(\tilde{w}) = \frac{1}{2} \sum (y_i - \tilde{y}_i)^2$
- SGD $\tilde{w}' = \tilde{w} - \eta \nabla J \tilde{x}_i$
  - $\nabla J = -(y_i - \tilde{y}_i)$
  - $\tilde{w}' = \tilde{w} + \eta (y_i - \tilde{y}_i) \tilde{x}_i$
Regression Tree

- Same structure as decision tree
- Predict = average target value or linear model at leaf (vs majority)
- Gain = reduction in standard deviation (vs entropy)

\[ \sigma = \sqrt{\sum (\tilde{y}_i - y_i)^2 / (N - 1)} \]
Rules

• Problem: very large decision trees have context that is complex and hard to understand

• Rules: self-contained, modular, easier to interpret, no need to cover universe

• $\mathcal{L}$ keeps sufficient statistics to:
  • make predictions
  • expand the rule
  • detect changes and anomalies
Adaptive Model Rules


- Ruleset: ensemble of rules
- Rule prediction: mean, linear model
- Ruleset prediction
  - Weighted avg. of predictions of rules covering instance \( x \)
  - Weights inversely proportional to error
  - Default rule covers uncovered instances

\[
\hat{f}(x) = \sum_{R_i \in S(x_i)} \theta_i \hat{y}_i,
\]

E.g: \( x = [4, -1, 1, 2] \)
Concept Drift
Definition

Given an input sequence \( \langle x_1, x_2, \ldots, x_t \rangle \), output at instant \( t \) an alarm signal if there is a distribution change, and a prediction \( \hat{x}_{t+1} \) minimizing the error \( |\hat{x}_{t+1} - x_{t+1}| \)

Outputs

- Alarm indicating change
- Estimate of parameter
Application

- Change detection on evaluation of model
- Training error should decrease with more examples
  - Change in distribution of training error
  - Input = stream of real/binary numbers
- Trade-off between detecting true changes and avoiding false alarms
Cumulative Sum

• Alarm when mean of input data differs from zero
• Memoryless heuristic (no statistical guarantee)
• Parameters: threshold $h$, drift speed $\nu$
• $g_0 = 0$, $g_t = \max(0, g_{t-1} + \varepsilon_t - \nu)$
• if $g_t > h$ then alarm; $g_t = 0$
Statistical Process Control


• Monitor error in sliding window

• Null hypothesis: no change between windows

• If error > warning level learn in parallel new model on the current window

• if error > drift level substitute new model for old
Concept-adapting VFDT

G. Hulten, L. Spencer, P. Domingos: “Mining Time-Changing Data Streams”. KDD ’01

- Model consistent with sliding window on stream
- Keep sufficient statistics also at internal nodes
  - Recheck periodically if splits pass Hoeffding test
  - If test fails, grow alternate subtree and swap-in when accuracy of alternate is better
- Processing updates $O(1)$ time, +$O(W)$ memory
  - Increase counters for incoming instance, decrease counters for instance going out window
Hoeffding Adaptive Tree

• Replace frequency counters by estimators
• No need for window of instances
• Sufficient statistics kept by estimators separately
• Parameter-free change detector + estimator with theoretical guarantees for subtree swap (ADWIN)
• Keeps sliding window consistent with “no-change hypothesis”
ADWIN

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

Problem

Given an input sequence \(x_1, x_2, \ldots, x_t, \ldots\) we want to output

- a prediction \(\hat{x}_{t+1}\) minimizing prediction error:

\[
|\hat{x}_{t+1} - x_{t+1}|
\]

- an alert if change is detected
ADWIN

Optimal Change Detector and Predictor

- High accuracy
- Fast detection of change
- Low false positives and false negatives ratios
- Low computational cost: minimum space and time needed

ADWIN

- Theoretical guarantees
- No parameters needed
ADWIN

Theorem

At every time step we have:

1. (False positive rate bound). If $\mu_t$ remains constant within $W$, the probability that ADWIN shrinks the window at this step is at most $\delta$.

2. (False negative rate bound). Suppose that for some partition of $W$ in two parts $W_0W_1$ (where $W_1$ contains the most recent items) we have $|\mu_{W_0} - \mu_{W_1}| > 2\epsilon_c$. 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.
ADWIN

● Classification
  ● Adaptive Naive Bayes (Bifet et al. 2007)
  ● Decision Trees: Hoeffding Adaptive Trees (Bifet et al. 2009)
  ● ADWIN Bagging (Bifet et al. 2009)
  ● Leveraging Bagging (Bifet et al. 2010)
  ● Stacking of Restricted Hoeffding Trees (Bifet et al. 2012)
  ● Multilabel Classification (Read et al. 2012)
  ● Adaptive kNN (Bifet et al. 2013)
  ● Random Forests (Marron et al. 2014)

● Frequent Pattern Mining
  ● Frequent Closed Tree Mining (Bifet et al. 2008)
  ● Frequent Closed Graph Mining (Bifet et al. 2011)
Adaptive Random Forest

- Why Random Forests?
  - Off-the-shelf learner
  - Good learning performance Related publication

**Adaptive random forests for evolving data stream classification.**

Gomes, H M; Bifet, A; Read, J; Barddal, J P; Enembreck, F; Pfharinger, B; Holmes, G; Abdessalem, T.


- Based on the original Random Forest by Breiman
Adaptive Random Forest

1. Simulates resampling through leveraging bagging
2. Randomly select subsets of features for splits
3. Uses Hoeffding Trees as the base learner
4. 1 drift and 1 warning detector per tree
5. Train trees in the background before adding them
6. Trees are completely independent (can train in parallel)
SAM-kNN

KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift.

Viktor Losing, Barbara Hammer, Heiko Wersing:

*Best Paper Award*
ICDM 2016: 291-300
Micro-Clusters

Tian Zhang, Raghu Ramakrishnan, Miron Livny: “BIRCH: An Efficient Data Clustering Method for Very Large Databases”. SIGMOD ’96

- AKA, Cluster Features CF
  Statistical summary structure

- Maintained in online phase, input for offline phase

- Data stream $\langle \bar{x}_i \rangle$, d dimensions

- Cluster feature vector
  $N$: number of points
  $LS_j$: sum of values (for dim. j)
  $SS_j$: sum of squared values (for dim. j)

- Easy to update, easy to merge

- **Constant space irrespective to the number of examples!**

Properties:

- Centroid $= \frac{LS}{N}$
- Radius $= \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2}$
- Diameter $= \sqrt{\frac{2 \times N \times SS - 2 \times LS^2}{N \times (N-1)}}$
MOA Algorithms

- Multi-label/ Multi-target
- Outlier Detection
- Concept Drift Detection
- Active Learning
- Frequent Itemset Mining
- Frequent Graph Mining
- Recommendation Systems
What’s next?
I1: Validation Methodology

Theory suggests k-fold
- Cross-validation
- Split-validation
- Bootstrap validation
I1: Validation Methodology

Theory suggests k-fold
- Cross-validation
- Split-validation
- Bootstrap validation
II: Validation Methodology

Theory suggests k-fold
- Cross-validation
- Split-validation
- Bootstrap validation

- Cross-validation strongest, but most expensive
- Split-validation weakest, but cheapest
- Bootstrap: in between, but closer to cross-validation
Evaluation can be misleading
“Magic” classifier

Electricity Dataset, Accuracy

Accuracy, %

Time, instances

- Magic Classifier
- VFDT
- Majority Class
- Naive Bayes
# Published results

## Electricity Dataset, Accuracy

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDM</td>
<td>89.6*</td>
</tr>
<tr>
<td>Learn++.CDS</td>
<td>88.5</td>
</tr>
<tr>
<td>KNN-SPRT</td>
<td>88.0</td>
</tr>
<tr>
<td>GRI</td>
<td>88.0</td>
</tr>
<tr>
<td>FISH3</td>
<td>86.2</td>
</tr>
<tr>
<td>EDDM-IB1</td>
<td>85.7</td>
</tr>
<tr>
<td><strong>Magic classifier</strong></td>
<td><strong>85.3</strong></td>
</tr>
<tr>
<td>ASHT</td>
<td>84.8</td>
</tr>
<tr>
<td>bagADWIN</td>
<td>82.8</td>
</tr>
<tr>
<td>DWM-NB</td>
<td>80.8</td>
</tr>
</tbody>
</table>

* tested on a subset

<table>
<thead>
<tr>
<th>Algorithm name</th>
<th>Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local detection</td>
<td>80.4</td>
</tr>
<tr>
<td>Perceptron</td>
<td>79.1</td>
</tr>
<tr>
<td>AUE2</td>
<td>77.3</td>
</tr>
<tr>
<td>ADWIN</td>
<td>76.6</td>
</tr>
<tr>
<td>EAE</td>
<td>76.6</td>
</tr>
<tr>
<td>Prop. method</td>
<td>76.1</td>
</tr>
<tr>
<td>Cont. $\lambda$-perc.</td>
<td>74.1</td>
</tr>
<tr>
<td>CALDS</td>
<td>72.5</td>
</tr>
<tr>
<td>TA-SVM</td>
<td>68.9</td>
</tr>
</tbody>
</table>
“Magic” = no-change classifier

- Problem is Auto-correlation
- Use for evaluation: Kappa-plus
- Exploit for better prediction
“Magic” = no-change classifier

Kappa Plus Statistic

- $p_0$: classifier’s prequential accuracy
- $p_e$: no-change classifier’s prequential accuracy

- $\kappa^+$ statistic
  \[
  \kappa^+ = \frac{p_0 - p_e}{1 - p_e}
  \]

- $\kappa^+ = 1$ if the classifier is always correct
- $\kappa^+ = 0$ if the predictions coincide with the correct ones as often as those of the no-change classifier
SWT: Temporally Augmented Classifier

\[ Pr[\text{class is } c] \equiv h(x^t, c^{t-\ell}, \ldots, c^{t-1}) \]
SWT: Accuracy and Kappa Plus, Electricity
SWT: Accuracy and Kappa Plus, Forest Cover
http://huawei-noah.github.io/streamDM-Cpp/

streamDM C++
G. De Francisci Morales, A. Bifet: “SAMOA: Scalable Advanced Massive Online Analysis”. JMLR (2014)
SAMOA ARCHITECTURE

- Classifier Methods
- Clustering Methods
- Frequent Pattern Mining

SAMOA

- S4
- Storm
- ...
- Flink
Vertical Partitioning

N. Kourtellis, G. De Francisci Morales, A. Bifet, A. Murdopo: “VHT: Vertical Hoeffding Tree”, 2016

Big Data Conference 2016

Stream

Model

Stats

Attributes

Stats

Stats

Splits

Single attribute tracked in single node
Kappa Architecture

- Apache Kafka is a fast, scalable, durable, and fault-tolerant publish-subscribe messaging system.
SUPPORTING ORGANISATIONS

- Yahoo! Labs
- Huawei
- Telefonica
- Telecom ParisTech
- Aalto University
- The University of Waikato
http://huawei-noah/github.io/streamDM

StreamDM

streamDM: Data Mining for Spark Streaming
Summary

• IoT Streaming useful for finding approximate solutions with reasonable amount of time & limited resources

• MOA: Massive Online Analytics
  • Available and open-source
    • http://moa.cms.waikato.ac.nz/

• SAMOA: A Platform for Mining Big Data Streams
  • Available and open-source (incubating @ASF)
    • http://samoa.incubator.apache.org
Open Challenges

- Times Series + Stream Mining
- Structured output
- Millions of classes
- Ease of use
- Applications: Predictive Maintenance, AI for IoT
Thanks!

@abifet