Data Stream Processing and Analytics

Vincent Lemaire

Thank to Alexis Bondu, EDF
Outline

• Introduction on data-streams
• Part 1 : Querying
• Part 2 : Unsupervised Learning
• Part 3 : Supervised Learning
• Conclusion
Outline

- Introduction on data-streams
- Part 1: Querying
- Part 2: Unsupervised Learning
- Part 3: Supervised Learning
- Conclusion
Big Data – what does that mean?

<table>
<thead>
<tr>
<th>Volume</th>
<th>From...</th>
<th>...to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal</td>
<td>Internal (terabyte)</td>
<td>External (petabyte)</td>
</tr>
<tr>
<td>Velocity</td>
<td>Batch</td>
<td>Real time</td>
</tr>
<tr>
<td>Variety</td>
<td>Structured</td>
<td>Unstructured</td>
</tr>
<tr>
<td>Visualization</td>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Value creation</td>
<td>Reporting and compliance</td>
<td>Strategic asset / Competitive advantage</td>
</tr>
</tbody>
</table>
Big Data Analytics?

- Big Data Analytics: Extracting Meaningful and Actionable Information from a Massive Source

- Let’s avoid
  - Triviality, Tautology: a series of self-reinforcing statements that cannot be disproved because they depend on the assumption that they are already correct
  - Thinking that noise is an information

- Let’s try to have
  - Translation: capacity to transfer in concrete terms the discovery (actionable information)
  - TTM: Time To Market, ability to have quickly information on every customers (Who, What, Where, When)
Big Data vs. Fast Data

- **Big Data**:
  - Static data
  - **Storage**: distributed on several computers
  - **Query & Analysis**: distributed and parallel processing
  - **Specific tools**: Very Large Database *(ex: Hadoop)*

- **Fast Data**:
  - Data in motion
  - **Storage**: none *(only buffer in memory)*
  - **Query & Analysis**: processing on the fly *(and parallel)*
  - **Specific Tools**: CEP *(Complex Event Processing)*

More than 10 To

More than 1000 operations / sec
Application Areas

- **Finance:** High frequency trading
  - Find **correlations** between the prices of stocks within the historical data;
  - Evaluate the **stationarity** of these correlations **over the time**;
  - Give more **weight to recent data**.

- **Banking:** Detection of frauds with credit cards
  - Automatically **monitor** a **large amount** of transactions;
  - **Detects patterns** of events that indicate a likelihood of fraud;
  - **Stop** the processing and **send an alert** for a human adjudication.

- **Medicine:** Health monitoring
  - Perform **automatic medical analysis** to reduce workload on nurses;
  - Analyze measurements of devices to **detect early signs** of disease;
  - Help doctors to make a **diagnosis** in real time.

- **Smart Cities & Smart grid:**
  - Optimization of **public transportation**;
  - Management of the **local production** of electricity;
  - Flattening of the **evening peak** of consumption.
An example of data stream

Input data stream

Online processing:
Rotate and combine tuples in a compact way

A tuple:
(1,1);(1,2);(2,2);(1,3)

All tuples can be coded by 4 couples of integers
Specific constrains of stream-processing

What is a tuple?
- A small **piece of information in motion**
- Composed by several variables
- All tuples share the **same structure** (i.e. the variables)

What is a data stream?
- A data stream **continuously emits** tuples
- The **order** of tuples is not controlled
- The emission **rate** of tuples is not controlled
- Stream processing is an **on-line process**

In the end, **the quality** of the processing is the **adjusting variable**
How to manage the time?

• A timestamp is associated with each tuple:
  – Explicit timestamp: defined as a variable within the structure of the data stream
  – Implicit timestamp: assigned by the system when tuples are processed

• Two ways of representing the time:
  – Logical time: only the order of processed tuples is considered
  – Physical time: characterizes the time when the tuple was emitted

• Buffer issues:
  – The tuples are not necessarily received in the order
  – How long a missing tuple can be waited?
Complex Events Processing (CEP)

- An operator implements a **query** or a more complex **analysis**
- An operator processes data in motion with a **low latency**
- Several operators run **at the same time**, parallelized on several CPUs and/or Computers
- The graph of operators is **defined before** the processing of data-streams
- Connectors allows to interact with: **external data streams, static data** in SGBD, **visualization** tools.
Complex Events Processing (CEP)

Main features:
- High frequency processing
- Parallel computing
- Fault-tolerant
- Robust to imperfect and asynchronous data
- Extensible *(implementation of new operators)*

Notable products:
- StreamBase *(Tibco)*
- InfoSphere Streams *(IBM)*
- STORM *(Open source – Twitter)*
- KINESIS *(Amazon)*
- SQLstream
- Apama
Outline

• Introduction on data-streams
• Part 1 : Querying
• Part 2 : Unsupervised Learning
• Part 3 : Supervised Learning
• Conclusion
Time-window

- A query is performed on a finite part of the past tuples

Define a time-window:
- Fixed window: “June 2000”
- Sliding window: “last week”
- Landmark window: “since 1 January 2000”

Update interval:
- A result is produced at each update
- The type of window depends on the update interval
Most of the CEP provide a SQL-like language

- Few CEP provide a user-friendly interface
- Each software publisher propose its own language (*not standardized*)

Main features:

- Define the **structure** of the **connection** of the data streams
- Define **time-windows** on data streams
- **Extend** the SQL language (*able to run SQL queries on relational data bases*)
- **Run queries** on data streams within time-windows

Additional functions:

- **Statistics** *(min, max, mean, standard deviation … etc)*
- **Math** *(trigonometry, logarithm, exponential … etc)*
- **String** *(regular expression, trim, substring … etc)*
- **Date** *(getDayType, getSecond, now … etc)*
A simple example with StreamBase:

```
CREATE INPUT STREAM InputStream(
    Compteur string(12),
    Type string(12),
    Souscription int,
    C_index int,
    Date timestamp
);

CREATE WINDOW OneMinuteWindow SIZE 60
    ADVANCE 60 ON Date;
```
A simple example with StreamBase: 

```
CREATE OUTPUT STREAM OutputStream;

SELECT firstval(Compteur) AS Compteur,
     lastval(C_index) - firstval(C_index) AS Conso
FROM InputStream[OneMinuteWindow]
INTO OutputStream;
```
SQL-like language

A simple example with StreamBase: User-friendly interface
Outline

• Introduction on data-streams
• Part 1 : Querying
• Part 2 : Unsupervised Learning
• Part 3 : Supervised Learning
• Conclusion
Online vs. Batch mode

What is unsupervised learning?
- Mining data in order to find new knowledge
- No idea about the expected result

Batch mode:
- An entire dataset is available
- The examples can be processed several times
- Weak constrain on the computing time
- The distribution of data does not change over time

Online processing:
- Tuples are emitted one by one
- Tuples are processed on the fly due to their high rate
- Real-time computing (low latency)
- The distribution of tuples changes over time (drift)
Summarizing data streams

Why we need to summarize data streams?

- The number of tuples in infinite ...
- Their emission rate is potentially very high ...
- The hardware resources are limited (CPU, RAM & I/O)

What is a summary?

- A compact representation of the past tuples
- With a controlled memory space, accuracy and latency
- Which allows to query (or analyze) the history of the stream, in an approximated way

The objective is to maximize the accuracy of the queries, given technical constrains (stream rate, CPU, RAM & I/O)
Two types of summary

Specific summaries: dedicated to a single query *(or few)*

- **Flajolet-Martin Sketch**: approximates the number of unique objects in a stream;
- **Bloom Filter**: efficiently tests if an element is a member of a predefined set;
- **Count-Sketch**: efficiently finds the $k$ most frequent elements of a set;
- **Count-Min Sketch**: enumerates the number of elements with a particular value, or within an interval of values.

Generic summaries: allow a large range of queries on any past period

- **StreamSamp**: based on successive windowing and sampling;
- **CluStream**: based on micro-clustering;
- **DenStream**: based on evolving micro-clustering;

Detailed in this talk
Flajolet-Martin Sketch [1] approximates the number of unique objects in a stream

**Problem statement:**

- S is a collection of N elements: $S = \{s_1, s_2 \ldots s_N\}$
- Two elements of S may be identical
- S includes only F distinct elements
- The objective is to efficiently estimate F in terms of:
  - Time complexity
  - Space complexity
  - Probabilistic guarantee
Hash function: $h(.)$

- Associates an element $s_i$ with a random binary value
- $h(.)$ is a deterministic function
- $w$ is the length of binary values (number of bits)
- $w$ is an integer such that $2^w \geq N \geq F$
- Random values are uniformly drawn within $0, 2^w − 1$

Intuition:

Given a large set of random binary values,

- $\frac{1}{2}$ of them begin with “1”
- $\frac{1}{4}$ of them begin with “11”
- $\frac{1}{8}$ of them begin with “111”
- $\frac{1}{2^k}$ of them begin with $k$ “1”
Flajolet-Martin Sketch [1]

Location of the first “1” within $h(.)$

t(.) is the function which keeps only the first “1” (counting from left), other bits are set to “0”

Example: $h(s_i) = 0100111011010$ $t(h(s_i)) = 0100000000000$

Fusion of binary words:

$B$ is the fusion of all the binary words $t(h(s_i))$ by using the OR operator denoted by $\oplus$

$$B = \bigoplus_{i=1}^{N} t(h(s_i))$$

$$R = \max_{i=1}^{N} t(h(s_i)) + 1$$

$R$ is the rank of the first “0” (counting from left) within $B$. That is a random variable related with $F$!
Flajolet-Martin Sketch [1]

A single-pass algorithm:

Input data stream

\[ h(a) = 01001111011010 \]

FM Sketch

\[ R = 0 \]
\[ B = 000000000000000 \]

\[ t(h(a)) = 0100000000000000 \]
Flajolet-Martin Sketch [1]

A single-pass algorithm:

Input data stream

\[
h(a) = 01001111011010 \quad t(h(a)) = 01000000000000
\]

\[
h(b) = 10001010011011 \quad t(h(b)) = 10000000000000
\]
Flajolet-Martin Sketch [1]

A single-pass algorithm:

Input data stream

\[ h(a) = 01001111011010 \quad t(h(a)) = 0100000000000000 \]
\[ h(b) = 10001010011011 \quad t(h(b)) = 1000000000000000 \]
\[ h(a) = 01001111011010 \quad t(h(a)) = 0100000000000000 \]

FM Sketch

\[ R = 2 \]
\[ B = 1100000000000000 \]
A single-pass algorithm:

Input data stream

\[ h(a) = 01001111011010 \]
\[ t(h(a)) = 01000000000000 \]
\[ h(b) = 10001010011011 \]
\[ t(h(b)) = 10000000000000 \]
\[ h(a) = 01001111011010 \]
\[ t(h(a)) = 01000000000000 \]
\[ h(c) = 00010110010110 \]
\[ t(h(c)) = 00010000000000 \]

FM Sketch
\[ R = 2 \]
\[ B = 110000000000000 \]
Flajolet-Martin Sketch [1]

A single-pass algorithm:

Input data stream

\[ h(a) = 01001111011010 \]
\[ t(h(a)) = 010000000000000 \]
\[ h(b) = 10001010011011 \]
\[ t(h(b)) = 100000000000000 \]
\[ h(a) = 01001111011010 \]
\[ t(h(a)) = 010000000000000 \]
\[ h(c) = 00010110010110 \]
\[ t(h(c)) = 000100000000000 \]
\[ h(b) = 10001010011011 \]
\[ t(h(b)) = 100000000000000 \]

FM Sketch
\[ R = 4 \]
\[ B = 110100000000000 \]
A single-pass algorithm:

- **Input data stream**

  - $h(a) = 01001111011010$ \quad $t(h(a)) = 010000000000000$
  - $h(b) = 10001010011011$ \quad $t(h(b)) = 100000000000000$
  - $h(a) = 01001111011010$ \quad $t(h(a)) = 010000000000000$
  - $h(c) = 00010110010110$ \quad $t(h(c)) = 000100000000000$
  - $h(b) = 10001010011011$ \quad $t(h(b)) = 100000000000000$

- This single-pass algorithm is adapted to **data streams**
- **Few pieces of information** need to be **stored** in the RAM
- **R** is a random variable such that:

$$E(R) \approx \log_2 \varphi F$$
How to estimate $E(R)$?

- The first bits of $h(s_i)$ are used to affect each element to a sketch.
  - If $m = 16$, the 4 first bits of $h(s_i)$ represent the ID of the corresponding Sketch.
  - $h(s_1) = 0011001000101$ -> $0011 = 3$ -> $s_1$ is affected to the 3th sketch.
- Each Sketch counts approximately $F/m$ distinct elements.

**Flajolet-Martin Sketch [1]**
Flajolet-Martin Sketch [1]

Input data stream → Deterministic rooting of $h(s_i)$ → Collection of $m$ Sketches

How to estimate $E(R)$?

$F \approx \frac{2^{E(R)}}{m}$

Stochastic average:

$F \approx \frac{m}{\varphi} \left( \frac{1}{m} \sum_{i=1}^{m} R_m \right)$
Two types of summary

Specific summaries: dedicated to a single query (or few)

- **Flajolet-Martin Sketch**: approximates the number of unique objects in a stream;
- **Bloom Filter**: efficiently tests if an element is a member of a predefine set;
- **Count-Sketch**: efficiently finds the $k$ most frequent elements of a set;
- **Count-Min Sketch**: enumerates the number of elements with a particular value, or within an interval of values.

Generic summaries: allow a large range of queries on any past period

- **StreamSamp**: based on successive windowing and sampling;
- **CluStream**: based on micro-clustering;
- **DenStream**: based on evolving micro-clustering;

Detailed in this talk
Objectives of a generic summary:

- Summarizes the entire history of the data stream
- Requires a bounded memory space
- Allows a large range of queries, including supervised and unsupervised analysis

Summarize by sampling the tuples:

- The sampling techniques are adapted to incremental processing
- A limited number of tuples are stored
- The stored tuples constitute a representative sample
- The recent past can be favored in terms of accuracy (i.e. sampling rate)
Sampling based summaries

Reservoir sampling [2]

- The reservoir is a uniform sampling;
- The sampling rate decreases over time;
- The probability that tuples are included in the reservoir is: \( \frac{k}{\text{Nb}_\text{Emitted}_\text{Tuples}} \)
Sampling based summaries

StreamSamp [3]

Input data stream

Uniform sampling

Sample 1

Tuple 1
Tuple 2
Tuple 3
Tuple 4

Order 0
Order 1
Order 2
Sampling based summaries

Input data stream

Uniform sampling

Sample 2
- Tuple 5
- Tuple 6
- Tuple 7
- Tuple 8

Sample 1
- Tuple 1
- Tuple 2
- Tuple 3
- Tuple 4

Order 0

Order 1

Order 2

StreamSamp [3]
Sampling based summaries

Input data stream

Uniform sampling

Sample 3
(Tuple 9
Tuple 10
Tuple 11
Tuple 12)

Sample 2
(Tuple 5
Tuple 6
Tuple 7
Tuple 8)

Sample 1
(Tuple 1
Tuple 2
Tuple 3
Tuple 4)

StreamSamp [3]

Order 0

Order 1

Order 2

Tuple 2
Tuple 3
Tuple 5
Tuple 8

Fusion

2
Sampling based summaries

Input data stream

Uniform sampling

StreamSamp [3]

Sample 4
- Tuple 13
- Tuple 14
- Tuple 15
- Tuple 16

Sample 3
- Tuple 9
- Tuple 10
- Tuple 11
- Tuple 12

Sample 1
- Tuple 2
- Tuple 3
- Tuple 5
- Tuple 8

Order 0

Order 1

Order 2
Sampling based summaries

Uniform sampling

Input data stream

StreamSamp [3]

Sample 5
- Tuple 17
- Tuple 18
- Tuple 19
- Tuple 20

Sample 4
- Tuple 13
- Tuple 14
- Tuple 15
- Tuple 16

Sample 3
- Tuple 9
- Tuple 10
- Tuple 11
- Tuple 12

Order 0

Sample 1
- Tuple 2
- Tuple 3
- Tuple 5
- Tuple 8

Order 1

Sample 2

Order 2
Sampling based summaries

StreamSamp [3]

Input data stream

Uniform sampling

Sample 5
(Tuple 17
Tuple 18
Tuple 19
Tuple 20)

Sample 4
(Tuple 13
Tuple 14
Tuple 15
Tuple 16)

Sample 3
(Tuple 9
Tuple 10
Tuple 11
Tuple 12)

Sample 1-
(Tuple 2
Tuple 3
Tuple 5
Tuple 8)

Sample 3-
(Tuple 9
Tuple 11
Tuple 14
Tuple 16)

Sample 2

2

Fusion

Order 0

Order 1

Order 2
Sampling based summaries

Input data stream

Uniform sampling

StreamSamp [3]

Order 0

Order 1

Order 2
Sampling based summaries

StreamSamp [3]

- A sample gathers $k$ uniformly drawn tuples
- A collection of samples gathers $h$ samples
- Each collection has an order $o$
- The sampling rate of samples is equal to $2^o$

(Past)  
Order 2  
Order 1  
Order 0  
(Present)

Time
Sampling based summaries

StreamSamp [3]

How to exploit this summary offline?

- Fusion of all samples
- Weighting of tuples to keep their representativeness

\[ W_{tuple} = \frac{2^o}{\text{weight}} \]

- Use of any *Datamining* approach able to process a weighted training set
Two types of summary

Specific summaries: dedicated to a single query (or few)

- Flajolet-Martin Sketch: approximates the number of unique objects in a stream;
- Bloom Filter: efficiently tests if an element is a member of a predefined set;
- Count-Sketch: efficiently finds the $k$ most frequent elements of a set;
- Count-Min Sketch: enumerates the number of elements with a particular value, or within an interval of values.

Generic summaries: allow a large range of queries on any past period

- StreamSamp: based on successive windowing and sampling;
- CluStream: based on micro-clustering;
- DenStream: based on evolving micro-clustering;
What is a micro-clustering?

- Micro-clusters (MC) are small groups of tuples,
- MC are represented by features which locally describe the density of tuples,
- DenStream: Micro-clustering approaches handle evolving data,
- MC are maintained in RAM memory within a bounded memory space,
- MC summarize the density of the input data stream, while giving more importance to the recent past.
Micro-clustering based summaries

DenStream [4]

- **Density based** micro-clustering
- **Weighting** of the tuples over time

\[ W = 2^{-\lambda t} \]
Micro-clustering based summaries

DenStream [4]

Initialization of the micro-clusters with DBscan

Parameters:
- Minimum weight of Mc
- Maximum variance of Mc
- Fading factor

MC(cj,rj,wj)
Micro-clustering based summaries

DenStream [4]
Micro-clustering based summaries

DenStream [4]

\[ W \leftarrow W . 2^{-\lambda \cdot \Delta t} \]

Fading of the micro-clusters
Micro-clustering based summaries

DenStream [4]
Micro-clustering based summaries

**DenStream [4]**

Test on the new variance
Micro-clustering based summaries

DenStream [4]

Here, the new variance is greater than the maximum variance.
Micro-clustering based summaries

**DenStream** [4]

An “outlier” micro-cluster is created
Micro-clustering based summaries

DenStream [4]
Micro-clustering based summaries

DenStream [4]

Fading of the micro-clusters
Micro-clustering based summaries

DenStream [4]

Closest micro-cluster
Micro-clustering based summaries

DenStream [4]

Test on the new variance
Micro-clustering based summaries

DenStream [4]

Here, the new variance is less than the maximum variance.
Micro-clustering based summaries

DenStream [4]

1) The tuple is assigned to the micro-cluster

2) The weight, the variance and the mean are updated
Micro-clustering based summaries

DenStream [4]
Micro-clustering based summaries

DenStream [4]

Fading of the micro-clusters
Micro-clustering based summaries

DenStream [4]
Micro-clustering based summaries

DenStream [4]

Test on the new variance, which is too important
Micro-clustering based summaries

**DenStream [4]**

Closest “outlier”
Micro-cluster
Micro-clustering based summaries

DenStream [4]

Test on the new variance
Micro-clustering based summaries

DenStream [4]

In this case, the tuple is assigned to the “outlier” micro-cluster.
Micro-clustering based summaries

DenStream [4]

The weight of the “outlier” micro-cluster is greater than the minimum weight
Micro-clustering based summaries

DenStream [4]

The “outlier” micro-cluster becomes a regular micro-cluster
Micro-clustering based summaries

DenStream [4]

How to exploit this summary to estimate the density of the data stream?

... the example of the Parzen windows estimator [5] ...

\[ \hat{P}(x) = \frac{1}{N} \sum_{i=1}^{N} K(x - x_i) \]

\[ K(x - x_i) = \frac{1}{(\sigma \sqrt{2\pi})^k} \exp\left(-\frac{d(x, x_i)^2}{2 \sigma^2}\right) \]
Micro-clustering based summaries

DenStream [4]

Adapted Parzen window [5]:

\[ \hat{P}^*(x) = \frac{1}{C \cdot W} \sum_{j=1}^{C} \frac{w_j}{\sqrt{2\pi (\delta_j^2 + r_j^2)^k}} \exp \left( -\frac{d(x, c_j)^2}{2(\delta_j^2 + r_j^2)} \right) \]

- \( W \) : total weight of the data stream
- \( C \) : number of micro-clusters
- \( w_j \) : weight of the \( j \)-th micro-cluster
- \( r_j \) : standard deviation of the \( j \)-th micro-cluster
- \( \delta \) : smoothing parameter

Hypothesis: each tuple represents a set of none-observed tuples, with a fixed effective and a standard deviation equal to

Law of total variance
Main ideas to retain:

- **Summaries** allow to process data streams with very high emission rate,
- By using **limited hardware resources** *(CPU, RAM)*.
- In most cases, a trade off must be reached between the **accuracy** and the available memory.
- There are **two types of summary** *(specific and generic)*
- **Limitation**: most of generic summaries involves user **parameters**.
References


Related documents:

Outline

- Introduction on data-streams
- Part 1: Querying
- Part 2: Unsupervised Learning
- Part 3: Supervised Learning
- Conclusion
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
What is supervised learning?

- **Output**: prediction of a **target variable** for new observations
- **Data**: a supervised model is **learned** from **labeled examples**
- **Objective**: learn **regularities** from the training set and **generalize** it *(with parsimony)*

Several types of supervised models:

- Categorical target variable -&gt; **Classifier**
- Numeric target variable -&gt; **Regression**
- Time series -&gt; **Forecasting**
From Batch mode to Online Learning

A learning algorithm exploits the training set to automatically adjust the classifier.
Batch mode learning:

- An entire dataset is available
- The examples can be processed several times
- Weak constrain on the computing time
- The distribution of data does not change

Any time learning algorithm:

- Can be interrupted before its end
- Returns a valid classifier at any time
- Is expected to find better and better classifier
- Relevant for time-critical application
From Batch mode to Online Learning

**Incremental learning algorithm:**
- Only a **single pass** on the training examples is required.
- The classifier is **updated** at each **example**.
- **Avoid** the **exhaustive storage** of the examples in the **RAM**.
- Relevant for **time-critical** applications and for **progressively recorded** data.

**Online learning algorithm:**
- The training set is substituted by an **input data stream**
- The classifier is **continually updated** over time,
- By exploiting the **current** tuple,
- With a very **low latency**.
- The **distribution** of data **can change** over time (concept drift)
From Batch mode to Online Learning

Machine Learning: What are the pros and cons of offline vs. online learning?

Try to find answers to:
(which is which)

• Computationally much faster and more space efficient
• Usually easier to implement
• A more general framework.
• More difficult to maintain in production.
• More difficult to evaluate online
• Usually more difficult to get "right".
• More difficult to evaluate in an offline setting, too.
• Faster and cheaper
• …
From Batch mode to Online Learning

Focus today - Supervised classifier

- Try to find answers to:
  - Can the examples be stored in memory?
  - Which is the availability of the examples: any presents? In stream? Visible only once?
  - Is the concept stationary?
  - Does the algorithm have to be anytime? (time critical)
  - What is the available time to update the model?
  - ...

- The answers to these questions will give indications to select the algorithms adapted to the situation and to know if one need an incremental algorithm, even a specific algorithm for data stream.
STREAM MINING IS REQUIRED... SOMETIMES
From Batch mode to Online Learning

but…

Do not make the confusion!

Between Online Learning

and Online Deployment

A lot of advantages and drawback for both – but offline learning used 99% of the time
From Batch mode to Online Learning

“Incremental / online learning”: a new topic?

The first learning algorithms were all incremental:

• Perceptron [Rosenblatt, 1957-1962]
• CHECKER [Samuel, 1959]
• ARCH [Winston, 1970]
• Version Space [Mitchell, 1978, 1982], ...

However, most existing learning algorithms are not!
From Batch mode to Online Learning

Why not use the classic algorithms?

Classic decision tree learners assume all training data can be simultaneously stored in main memory

From Batch mode to Online Learning

Stream - supervised classification: what changes?

- Properties
  - Receives examples one-by-one
  - discards the example after processing it.
  - Produce a hypothesis after each example is processed
    - i.e. produces a series of hypotheses
  - No distinct phases for learning and operation
    - i.e. produced hypotheses can be used in classification
  - Allowed to store other parameters than model parameters (e.g. learning rate)
  - Is a real time system
    - Constraints: time, memory, ...
    - What is affected: hypotheses prediction accuracy
  - Can never stop
  - No i. i. d
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Implementation of on-line classifiers

Input stream: explicative variables
Output stream: predicted labels
Implementation of on-line classifiers

Second input stream: Real labels
Implementation of on-line classifiers
Implementation of on-line classifiers

In practice, this input stream may be delayed.

A on-line classifier predicts the class label of tuples before receiving the true label …
Implementation of on-line classifiers

Example: online advertising targeting

- **Input tuples**: couples “User – Ad”
- **Out tuples**: estimated probability that a User clicks on an Ad
Implementation of on-line classifiers

Example: online advertising targeting

User → Online Classifier → AgrMax(Ads) → Browser

P(Click) → Sending the Ad → Waiting for a click
Implementation of on-line classifiers

Example: online advertising targeting

User

Online Classifier

Update

Real labels

If clicked

Sending the Ad

Waiting for a click

After a fixed delay

P(Ad clicked)
Implementation of on-line classifiers

- Two streams exist
- Two drift detection have to be managed

- Labeled Data stream
  - $X_1, X_2, X_3, X_4, X_5, \ldots$
  - $0, 1, 1, 0, 0, \ldots$

- Unlabeled data stream
  - $X_1, X_2, X_3, X_4, X_5, \ldots$
  - $?, ?, ?, ?, ?, \ldots$

- Deployment
  - Predicted labels
    - $X_1, X_2, X_3, X_4, X_5, \ldots$
    - $1, 0, 1, 1, 0, \ldots$

- Models over the time
  - $f: X \rightarrow C$
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Evaluation of on-line classifiers

A – Holdout Evaluation

The stream of labeled tuples is split

Update

Evaluation on the recent past

Use of standard evaluation criteria
(Accuracy, BER, Lift curve, AUC … etc.)

Unbiased evaluation
Evaluation of on-line classifiers

B – Prequential Evaluation

Each labeled tuples is used twice

2 - Update

1 - Update

Online Classifier

On-line Evaluation

$X \rightarrow \hat{Y} \rightarrow Y$

From the beginning of the stream

$S = \sum_{i=1}^{n} L(y_i, \hat{y}_i)$

On the recent past

(buffer on a sliding window)
Evaluation of on-line classifiers

C – Kappa Statistic

- \( p_0 \): prequential accuracy of the classifier
- \( p_c \): probability that a random classifier makes a correct prediction.

\[
K = \frac{(p_0 - p_c)}{(1 - p_c)}
\]

- \( K = 1 \) if the classifier is always correct
- \( K = 0 \) if the predictions coincide with the correct ones as often as those of the random classifier
Evaluation of on-line classifiers

RAM Hours

A server RAM hour is the amount of RAM allocated to a server multiplied by the number of hours the server has been deployed.

Example: One 2 GB server deployed for 1 hour = 2 server RAM hours.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier A</td>
<td>70%</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Classifier B</td>
<td>80%</td>
<td>20</td>
<td>40</td>
</tr>
</tbody>
</table>
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Taxonomy of classifier for data stream

**full example memory** Store *all* examples
- allows for efficient restructuring
- good accuracy
- huge storage needed
Examples: ID5, ID5R, ITI

**no example memory** Only store statistical information in the nodes
- loss of accuracy (depending on the information stored or again huge storage needed)
- relatively low storage space
Examples: ID4

**partial example memory** Only store *selected* examples
- trade of between storage space and accuracy
Examples: FLORA, AQ-PM
Taxonomy of classifier for data stream

Model Management
- Number
- Granularity
- Weights

Data Management
- Full Memory
  - Weighting
  - Aging
- Partial Memory
  - Windowing
    - Fixed Size Windows
    - Weighting
    - Aging
    - Adaptive Size Window
    - Weighting
    - Aging
    - "No memory"

Detection
- Monitoring of performances
- Monitoring of properties of the classification model
- Monitoring of properties of the data

Adaptation
- Blind methods
- 'Informed methods'

It is necessary to adapt the classifier to the application context
Taxonomy of classifier for data stream

Incremental Algorithm (no stream)

- Decision Tree
  - ID4 (Schlimmer - ML’86)
  - ID5/ITI (Utgoff – ML’97)
  - SPRINT (Shaffer - VLDB’96)
  - ...
- Naive Bayes
  - Incremental (for the standard NB)
  - Learn fastly with a low variance (Domingos – ML’97)
  - Can be combined with decision tree: NBTree (Kohavi – KDD’96)
Taxonomy of classifier for data stream

Incremental Algorithm (no stream)

- Neural Networks
  - IOLIN (Cohen - TDM’04)
  - learn++ (Polikar - IJCNN’02),…
- Support Vector Machine
  - TSVM (Transductive SVM – Klinkenberg IJCAI’01),
  - PSVM (Proximal SVM – Mangasarian KDD’01),…
  - LASVM (Bordes 2005)
- Rules based systems
  - AQ15 (Michalski - AAAI’86), AQ-PM (Maloof/Michalski - ML’00)
  - STAGGER (Schlimmer - ML’86)
  - FLORA (Widmer - ML’96)
Taxonomy of classifier for data stream

Incremental Algorithm (for stream)

- **Rules**
  - FACIL (Ferrer-Troyano – SAC’04,05,06)
- **Ensemble**
  - SEA (Street - KDD’01) based on C4.5
- **K-nn**
  - ANNCAD (Law – LNCS‘05).
  - IBLS-Stream (Shaker et al – Evolving Systems” journal 2012)
- **SVM**
  - CVM (Tsang – JMLR’06)
Taxonomy of classifier for data stream

Incremental Algorithm (for stream)

- Decision Tree – the only ones used?
  - Domingos: VFDT (KDD’00), CVFDT (KDD’01)
  - Gama: VFDTc (KDD’03), UFFT (SAC’04)
  - Kirkby: Ensemble d’Hoeffding Trees (KDD’09)
  - del Campo-Avila: IADEM (LNCS’06)
Taxonomy of classifier for data stream

Properties of an efficient algorithm

- low and constant duration to learn from the examples;
- read only once the examples in their order of arrival;
- use of a quantity of memory fixed “a priori;”
- production of a model close to the “offline model”
- (anytime)
- concept drift management

(0) Domingos, P. et G. Hulten (2001). Catching up with the data: Research issues in mining data streams. In Workshop on Research Issues in Data Mining and Knowledge Discovery.
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Definitions

- A classification problem is defined as:
  - \( N \) is a set of training examples of the form \((x, y)\)
  - \( x \) is a vector of \( d \) attributes
  - \( y \) is a discrete class label
- Goal: To produce from the examples a model \( y = f(x) \) that predict the classes \( y \) for future examples \( x \) with high accuracy
Decision Tree Learning

- One of the most effective and widely-used classification methods

- Induce models in the form of decision trees
  - Each node contains a test on the attribute
  - Each branch from a node corresponds to a possible outcome of the test
  - Each leaf contains a class prediction
  - A decision tree is learned by recursively replacing leaves by test nodes, starting at the root
Incremental Decision Tree

The example of the Hoeffding Trees [D]

How an incremental decision trees is learned?

- Single pass algorithm,
- With a low latency,
- Which avoids the exhaustive storage of training examples in the RAM.
- The drift is not managed

Training examples are processed one by one

<table>
<thead>
<tr>
<th>Var 1</th>
<th>Var 2</th>
<th>...</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>12</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>Y</td>
<td>98</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>Y</td>
<td>4</td>
<td></td>
<td>A</td>
</tr>
</tbody>
</table>
Incremental Decision Tree

The 4 elements of an online tree

- Online decision tree:
  - a bound...
  - a split criterion
  - summaries in the leaves
  - a local model
Incremental Decision Tree

The 4 elements of an online tree

- Online decision tree:
  - a bound: *How many examples before cutting an attribute?*
  - a split criterion: *Which attribute and which cut point?*
  - summaries in the leaves: *How to manage high speed data streams?*
  - a local model: *How to improve the classifier?*
The 4 elements of an online tree

- Online decision tree:
  - a bound...
  - a split criterion
  - summaries in the leaves
  - a local model
Key ideas:
The best attribute at a node is found by exploiting a small subset of the labeled examples that pass through that node:

- The first examples are exploited to choose the root attribute
- Then, the other examples are passed down to the corresponding leaves
- The attributes to be split are recursively chosen …

✓ The Hoeffding bound answers the question: How many examples are required to split an attribute?
Hoeffding Bound

- Consider a random variable $a$ whose range is $R$
- Suppose we have $n$ observations of $a$
- Mean: $\bar{a}$
- Hoeffding bound states:

With probability $1 - \delta$, the true mean of $a$ is at least $\bar{a} - \varepsilon$

where

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$
Incremental Decision Tree

How many examples are enough?

- Let $G(X_i)$ be the heuristic measure used to choose test attributes (e.g. Information Gain, Gini Index).
- $X_a$: the attribute with the highest attribute evaluation value after seeing $n$ examples.
- $X_b$: the attribute with the second highest split evaluation function value after seeing $n$ examples.
- Given a desired $\delta$, if $\Delta G = \overline{G}(X_a) - \overline{G}(X_b) > \varepsilon$ after seeing $n$ examples at a node,
  - Hoeffding bound guarantees the true $\Delta G \geq \Delta \overline{G} - \varepsilon > 0$, with probability $1 - \delta$.
  - This node can be split using $X_a$, the succeeding examples will be passed to the new leaves.

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$
The example of the Hoeffding Trees [D]

The algorithm

This algorithm has been adapted in order to manage concept drift [E]
✓ By maintaining an incremental tree on a sliding windows
✓ Which allows to forget the old tuples
✓ A collection of alternative sub-trees is maintained in memory and used in case of drift
Incremental Decision Tree

An example of Hoeffding Tree: VFDT (Very Fast Decision Tree)

- A decision-tree learning system based on the Hoeffding tree algorithm
- Split on the current best attribute ($\delta$), if the difference is less than a user-specified threshold ($T$)
  - Wasteful to decide between identical attributes
- Compute $G$ and check for split periodically ($n_{\text{min}}$)
- Memory management
  - Memory dominated by sufficient statistics

“Mining High-Speed Data Streams”, KDD 2000. Pedro Domingos, Geoff Hulten
Experiment Results (VFDT vs. C4.5)

- Compared VFDT and C4.5 (Quinlan, 1993)
- Same memory limit for both (40 MB)
  - 100k examples for C4.5
- VFDT settings: $\delta = 10^{-7}$, $T=5\%$, $n_{\min}=200$
- Domains: 2 classes, 100 binary attributes
- Fifteen synthetic trees 2.2k – 500k leaves
- Noise from 0% to 30%
Incremental Decision Tree

Experiment Results

Accuracy vs. # examples

Accuracy as a function of the number of training examples
Incremental Decision Tree

Experiment Results

Tree size vs. # examples

Tree size as a function of number of training examples
An example of Hoeffding Tree in case of concept drift: CVFDT

- CVFDT (Concept-adapting Very Fast Decision Tree learner)
  - Extend VFDT
  - Maintain VFDT’s speed and accuracy
  - Detect and respond to changes in the example-generating process

- See the Part “Concept Drift” of this talk
The 4 elements of an online tree

- Online decision tree:
  - a bound…
  - a split criterion
  - summaries in the leaves
  - a local model
**Different Split Criterion**

- Used to transform a leaf into a node
  - determine at the same time on
    - which attribute to cut and
    - on which value (cut point).

- Uses the information contained in the summaries:
  - not on all data
  - a definitive action

- Batch algorithm used:
  - Gain ratio using entropy (C4.5)
  - Gini (CART)
  - MODL Level
A criterion for attribute selection

- Which is the best attribute?
  - The one which will result in the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Popular *impurity criterion*: *information gain*
  - Information gain increases with the average purity of the subsets that an attribute produces
  - Information gain uses entropy $H(p)$
- Strategy: choose attribute that results in greatest information gain
Incremental Decision Tree

Which attribute to select?
Consider entropy $H(\rho)$

Incremental Decision Tree

pure, 100% yes
not pure at all, 40% yes

pure, 100% yes
not pure at all, 40% yes

allmost 1 bit of information required to distinguish yes and no
Entropy: \( H(p) = -p \log(p) - (1-p) \log(1-p) \)

- \( H(0) = 0 \)  pure node, distribution is skewed
- \( H(1) = 0 \)  pure node, distribution is skewed
- \( H(0.5) = 1 \) mixed node, equal distribution

\[ \text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n \]
Incremental Decision Tree

Example: attribute “Outlook”

- “Outlook” = “Sunny”:
  \[
  \text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5 \log(2/5) - 3/5 \log(3/5) = 0.971 \text{ bits}
  \]

- “Outlook” = “Overcast”:
  \[
  \text{info}([4,0]) = \text{entropy}(1,0) = -1 \log(1) - 0 \log(0) = 0 \text{ bits}
  \]

- “Outlook” = “Rainy”:
  \[
  \text{info}([3,2]) = \text{entropy}(3/5,2/5) = -3/5 \log(3/5) - 2/5 \log(2/5) = 0.971 \text{ bits}
  \]

- Expected information for “Outlook”:
  \[
  \text{info}([3,2],[4,0],[3,2]) = (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971
  = 0.693 \text{ bits}
  \]

Note: \( \log(0) \) is not defined, but we evaluate \( 0 \times \log(0) \) as zero.
Computing the information gain

- Information gain:
  (information before split) – (information after split)

\[
\text{gain}("Outlook") = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693 = 0.247 \text{ bits}
\]

- Information gain for attributes from weather data:

\[
\begin{align*}
\text{gain}("Outlook") &= 0.247 \text{ bits} \\
\text{gain}("Temperature") &= 0.029 \text{ bits} \\
\text{gain}("Humidity") &= 0.152 \text{ bits} \\
\text{gain}("Windy") &= 0.048 \text{ bits}
\end{align*}
\]
Continuing to split

Incremental Decision Tree

\[
\text{gain}("Temperature") = 0.571 \text{ bits}
\]

\[
\text{gain}("Windy") = 0.020 \text{ bits}
\]

\[
\text{gain}("Humidity") = 0.971 \text{ bits}
\]
Incremental Decision Tree

The final decision tree

- Note: not all leaves need to be pure; sometimes identical instances have different classes
  - Splitting stops when data can’t be split any further
Highly-branching attributes

- Problematic: attributes with a large number of values (extreme case: customer ID)
- Subsets are more likely to be pure if there is a large number of values
  - Information gain is biased towards choosing attributes with a large number of values
  - This may result in overfitting (selection of an attribute that is non-optimal for prediction)
Gain ratio

- *Gain ratio*: a modification of the information gain that reduces its bias on high-branch attributes

- Gain ratio should be
  - Large when data is evenly spread
  - Small when all data belong to one branch

- Gain ratio takes number and size of branches into account when choosing an attribute
  - It corrects the information gain by taking the *intrinsic information* of a split into account (i.e. how much info do we need to tell which branch an instance belongs to)
The 4 elements of an online tree

- Online decision tree:
  - a bound…
  - a split criterion
  - summaries in the leaves
  - a local model
Summaries in the leaves

- Numerical attributes
  - Exhaustive counts [Gama2003]
  - Partition Incremental Discretization [Gama2006]
  - VFML: intervals defined by first values and used as cut points [Domingos]
  - Gaussian approximation [Pfahringer2008]
  - Quantiles based summary [GK2001]

- Categorical attributes
  - for each categorical variable and for each value the number of occurrences is stored (but CMS could be used)
The 4 elements of an online tree

- Online decision tree:
  - a bound…
  - a split criterion
  - summaries in the leaves
  - a local model
Local model

- Purpose: improve the quality of the tree (especially at the beginning of training)

- A good local model for online decision trees has to:
  - consume a small amount of memory
  - be fast to build
  - be fast to return a prediction

- A study on the speed (in number of examples) of different classifiers show that

  ➔ naive Bayes classifier has these properties

VFDT -> VFDTc
Incremental Decision Tree

Local model: naive Bayes classifier

- to predict the class it requires an estimation of the class conditional density, for every attribute $j$, $P(V_j|C)$:

$$P(C_z|x_k) = \frac{P(C_z) \prod_{j=1}^{J} P(V_j = x_{jk}|C_z)}{\sum_{t=1}^{C} \left[ P(C_t) \prod_{j=1}^{J} P(V_j = x_{jk}|C_t) \right]}$$
Incremental Decision Tree

Experimentations: Influence of the local model

![Graphs showing the influence of the local model on Accuracy and Random RBF with different examples.](image)
Incremental Decision Tree

Experimentations: Influence of the local model
Incremental Decision Tree

The 4 elements of an online tree

- Online decision tree:
  - a bound…
  - a split criterion
  - summaries in the leaves
  - a local model

Note: Summaries are used by the split criterion and the local model.

Idea: Try to have these 3 ‘coherent’
2 EXAMPLES
HOEFFDING TREE, NAÏVE BAYES
Incremental Naïve Bayes

Bayes’ Rule

\[
P(C, X) = P(C \mid X)P(X) = P(X \mid C)P(C)
\]

\[
P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)}
\]
Naive Bayes Classifiers

Task: Classify a new instance $D$ based on a tuple of attribute values $D = \langle x_1, x_2, \ldots, x_n \rangle$ into one of the classes $c_j \in C$

$$c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c_j)P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$
Naïve Bayes Classifier:
Naïve Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.

Naïve Bayes Conditional Independence Assumption:

- Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$. 
**Principe**: hypothèse d’indépendance conditionnelle des variables explicatives entre elles

**Point fort**: prédicteur très simple à calculer à partir des estimations univariées et des probabilités a priori des modalités cible

**Limites**:
- dégradation des performances lorsque les variables sont redondantes
- peu interprétable pour un grand nombre de variables
Incremental Naïve Bayes

\[
P_{NB}(Y = C \mid X = x^n) = \frac{P(Y = C) \prod_{k=1}^{K} p(x_i^n \mid C)}{\sum_{j=1}^{J} P(C_j) \prod_{k=1}^{K} p(x_i^n \mid C_j)}
\]

- Each instance, \( x_k \), is a vector of values (numerical or categorical)
- However, when the \( X_i \) are continuous we must choose some other way to represent the distributions \( P(X_i \mid Y) \).
  - discretization / grouping respectively for numerical / categorical variables
  - using a discretization method and a grouping method.
Well performing methods for supervised discretization

- **MDLP**: find the best intervals based on the entropy. The best number of interval is found using a MDL approach.

- **MODL**: based on Bayesian formalism and MDL principle. This method aims to find the best discretization parameters (intervals number, intervals boundaries, classes distribution within an interval) in a Bayesian way.
Related works – DBMS community

Online statistics

Goal: find the best execution plan

- **Reservoir** (kind of « reservoir sampling ») + « EqualFrequency histogram »
  Gibbons P, Matias Y, Poosala V. *Fast incremental maintenance of approximate histograms*. ACM Transactions on Database. 2002

- **Quantiles**: many quantiles lists are maintained. If memory become full some lists are merged to recover it.
  Manku GS, Rajagopalan S, Lindsay BG. *Approximate medians and other quantiles in one pass and with limited memory*. SIGMOD’98

- **Quantiles**: a data structure is used to maintain online ranks and errors. This method has strong error guarantee on the quantiles
  Greenwald M, Khanna S. *Space-efficient online computation of quantile summaries*. SIGMOD’01
Incremental Naïve Bayes

Related works – Data mining
Incremental discretization

- **IFFD**: Incremental Flexible Frequency Discretization. Keep all the data and adapt interval sizes between a minimum and a maximum

- **PID**: two levels discretization
  level 1: mix between “EqualFreq” and “EqualWidth”
  level 2: all batch methods

- **Gaussian approximation**: approximate the data distribution with a Gaussian per class: \( \mu \) and \( \sigma \) parameters are kept online.
  Very low memory footprint.
### Incremental Naïve Bayes

#### Methods comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Global / local</th>
<th>Multi variate</th>
<th>Parametric</th>
<th>Supervised</th>
<th>Online / stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal Width</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Equal Freq</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Greenwald Khanna</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>K-means clustering</td>
<td>Global and local</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes / No</td>
</tr>
<tr>
<td>PID (Layer 1)</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MDLP / MODL</td>
<td>Global and local</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IFFD</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes / No</td>
</tr>
<tr>
<td>Gaussian</td>
<td>Global</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3 criteria were proposed by: Dougherty J, Kohavi R, Sahami M. *Supervised and unsupervised discretization of continuous features*. ML1995.
Incremental Naïve Bayes

Online Discretization: Gaussian approximation

- **Gaussian Approximation (GAUSS)**
  - Assume values conform to Normal Distribution
  - Maintain five numbers (eg mean, variance, weight, max, min)
  - Note: not sensitive to data order
  - Incrementally updateable
  - Using the max, min information per class – split the range into N equal parts
Online Discretization: A two levels discretization

- **Level 1**: Greenwald et Khanna - GK (or another method adapted to streams) based on a quantile summary
  
  global / not supervised / parametric / online

- **Level 2**: MODL or MDLP methods based on the entropy for intervals quality and on MDL principle to stop finding new intervals
  
  global / supervised / without parameters

→ Both levels are based on order statistics
Incremental Naïve Bayes

Averaging of Naïve Bayes Classifier

\[
P(C_z|x_k) = \frac{P(C_z) \prod_{j=1}^{J} P(V_j = x_{jk}|C_z)}{\sum_{t=1}^{C} \left[ P(C_t) \prod_{j=1}^{J} P(V_j = x_{jk}|C_t) \right]}^W_j
\]

\[
P(C_z|x_k) = \frac{P(C_z) \prod_{j=1}^{J} P(V_j = x_{jk}|C_z)^W_j}{\sum_{t=1}^{C} \left[ P(C_t) \prod_{j=1}^{J} P(V_j = x_{jk}|C_t)^W_j \right]}^W_j
\]


Incremental Naïve Bayes

Averaging of Naïve Bayes Classifier

\[
P(C_z | x_k) = \frac{P(C_z) \prod_{j=1}^{J} P(V_j = x_{jk} | C_z)}{\sum_{t=1}^{C} \left[ P(C_t) \prod_{j=1}^{J} P(V_j = x_{jk} | C_t) \right]} 
\]

\[
P(C_z | x_k) = \frac{P(C_z) \prod_{j=1}^{J} P(V_j = x_{jk} | C_z) \times W_j}{\sum_{t=1}^{C} \left[ P(C_t) \prod_{j=1}^{J} P(V_j = x_{jk} | C_t) \times W_j \right]} 
\]

**Littérature**: les poids sont obtenus suite à un moyennage de modèles qui correspondent à des sélections de variables différentes (Hoeting et al., 1999) (Boullé, 2007)
Incremental Naïve Bayes

Wj? Intuition

on paper board
Incremental Naïve Bayes

Benefits of Averaging of Naïve Bayes Classifier

Mean of the ACC, AUC evaluation criteria on the 30 UCI data sets.

Methods to compute the weights

No weights

Same results conclusion on the large scale learning challenge

The “classic” averaging of naïve Bayes classifier requires the storage of all the data (a data table allowing the link between the instances and their labels).
On cherchera la pondération $w$ qui minimise la log-vraisemblance régularisée.
Incremental Naïve Bayes

Régularisation de la log-vraisemblance :

\[ f(w) = \sum_{k=1}^{K} w_k^p \]

parcimonie
• \( p > 1 \) : convexe mais non parcimonieux
• \( p \leq 1 \) : non convexe mais parcimonieux

Exemple : \( X_1, X_2 \) deux variables identiques

<table>
<thead>
<tr>
<th></th>
<th>( w_1 = w_2 = 0.5 )</th>
<th>( w_1 = 0; w_2 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1^2 + w_2^2 )</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Non parcimonieux</td>
<td></td>
</tr>
<tr>
<td>( \sqrt{w_1} + \sqrt{w_2} )</td>
<td>1.41</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parcimonieux</td>
</tr>
</tbody>
</table>
Incremental Naïve Bayes

Averaging of Naïve Bayes Classifier – Performances
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
The input stream is **not stationary**
- The distribution of data **changes** over time
- Two strategies: **adaptive learning** or **drift detection**
- Several types of concept drift:

\[ P(x,y) = P(x) \cdot P(y|x) \]

**Concept drift**

**Virtual drift** [B] (or **covariate shift**)

**Original data**

**Concept drift** [A]
Concept drift

What kinds of drift can be expected [C]?

- **Abrupt**
- **Gradual**
- **Incremental**
- **Reoccurring**

- Drift detection
- On-line adaptive learning
- Drift detection & models management
Some specific constraints to manage:

- Adapt to concept drift asap
- Distinguish noise from changes (*Robust to noise, Adaptive to changes*)
- Recognizing and reacting to reoccurring contexts
- Adapting with limited hardware resources (*CPU, RAM, I/O*)
Concept drift

Manage Drift?

- Either detect and:
  1) Retrain the model
  2) Adapt the current model
  3) Adapt statistics (summaries) on which the model is based
  4) Work with a sequence of
     - models
     - summaries
  - or detect anything but train (learn) fastly
    - a single models
    - an ensemble of models
Desired Properties of a System To Handle Concept Drift

- Adapt to concept drift asap
- Distinguish noise from changes
  - robust to noise, but adaptive to changes
- Recognizing and reacting to reoccurring contexts
- Adapting with limited resources
  - time and memory
Concept drift
Adaptive learning strategies

change detection and a follow up reaction

reactive, forgetting

Single classifier

Triggering
variable windows

Detectors

evolving

Forgetting
fixed windows, Instance weighting

Contextual
dynamic integration, meta learning

Dynamic ensemble
adaptive combination rules

maintain some memory

Ensemble

PAKDD-2011 Tutorial,
May 27, Shenzhen, China

A. Bifet, J.Gama, M. Pechenizkiy, I.Zliobaite
Handling Concept Drift: Importance, Challenges and Solutions
Concept drift
Adaptive learning strategies

Single classifier

Triggering
Detectors

Evolving
Forgetting
fixed windows, Instance weighting

Ensemble

Contextual

Dynamic ensemble

forget old data and retrain at a fixed rate
Concept drift
Adaptive learning strategies

Triggering:
- Detect a change and cut

Detectors:
- Variable windows

Evolving:
- Forgetting

Single classifier
- Ensemble

Contextual
- Dynamic ensemble
Concept drift
Adaptive learning strategies

- Single classifier
  - Detectors
  - Forgetting
- Ensemble
  - Contextual
  - Dynamic ensemble
    - Adaptive combination rules
    - Build many models, dynamically combine
Concept drift
Adaptive learning strategies

- **Triggering**
  - Single classifier
  - Detectors

- **Evolving**
  - Forgetting
  - Ensemble
    - Contextual (dynamic integration, meta learning)
    - Dynamic ensemble

build many models,
switch models according
to the observed incoming data
Concept drift

Which approach to use?

• changes occur over time
• we need models that evolve over time
• choice of technique depends on
  – what type of change is expected
  – user goals/ applications
Concept drift

Drift detection

General schema:

- **Fixed Classifier** *(applied online)*
  - $X \rightarrow \hat{Y}$

- **Drift Detection**
  - $X \rightarrow Y \rightarrow \hat{Y}$

  - If detected:
    - Replace the classifier
    - Train a new classifier on the recent past
    - Adapt the size of the history
How to detect the drift?

Based on the online evaluation:

- **Main idea**: if the performance of the classifier changes, that means a drift is occurring ...
- **For instance**: if the error rate increases, the size of the sliding windows decreases and the classifier is retrained [F].
- **Limitation**: the user has to define a threshold
How to detect the drift?

Based on the distribution of tuples:

- **Main idea**: if the distributions of the “current window” and the “reference window” are significantly different, that means a drift is occurring ....
How to detect the drift?

Based on the distribution of tuples:

Detection of covariate shift: $P(X)$
- In [G] the author uses **statistical tests** in order to compare the both distributions
  - Welch test – *Mean values are the same?*
  - Kolmogorov Smirnov test – *Both samples of tuples come from the same distribution?*
- A **classifier** can be exploited to **discriminate** tuples belonging to **both windows** [H]
  - If the quality of the classifier is good, that means a drift is occurring …
  - Explicative variables: $X$
  - Target variable: $W$ (*the window*)

Detection of concept shift: $P(Y|X)$
- In [I] a classifier is exploited, the **class value** is considered as an **additional input variable**
  - Explicative variables: $X$ and $Y$
  - Target variable: $W$ (*the window*)
Handling Concept Drift: Importance, Challenges & Solutions

A. Bifet, J. Gama, M. Pechenizkiy, I. Žliobaitė

PAKDD-2011 Tutorial
May 27, Shenzhen, China

Concept drift

Parameters – The devil inside
Concept drift

No drift assumption?

Do not use online learning!
Outline

1. From Batch mode to Online Learning
2. Implementation of on-line classifiers
3. Evaluation of on-line classifiers
4. Taxonomy of classifier for data stream
5. Two examples
6. Concept drift
7. Make at simplest
Make at simplest!
(the first thing to test, the baseline)

Model Management

Full Memory
Weighting
Aging
Partial Memory
Windowing
Fixed Size Windows
Weighting
Aging
Adaptive Size Window
Weighting
Aging
"No memory"

Data Management

Detection

Monitoring of performances
Monitoring of properties of the classification model
Monitoring of properties of the data

Blind methods

'Informed methods'

Adaptation

Number
Granularity
Weights
A classifier trained with few examples but often!

- Which classifier?
  
  - Generative classifiers are better than discriminant classifiers when the number of examples is low and there is only one classifier (Bouchard 2004)
  
  - Ensemble of classifiers are very good (Bauer 1999)
  
  - Bagging of discriminative classifiers supplants a single generative classifier (and with a low variance) (Breiman 1996)
  
  - Methods "very" regularized "are very (too) strong (Cucker 2008)
A classifier trained with few examples but often!

- **Which classifier?**
  - a random forest (based on « *Learning with few examples: an empirical study on leading classifiers*, Christophe Salperwyck and Vincent Lemaire, in International Joint Conference on Neural Networks (IJCNN July 2011)»)
  - using 4096 examples
Make at simplest

Waveform
Make at simplest

Waveform
Make at simplest

Waveform
Make at simplest

Alternative problem settings
Alternative problem settings

Multi-armed bandits explore and exploit online set of decisions, while minimizing the cumulated regret between the chosen decisions and the optimal decision.

Originally, Multi-armed bandits have been used in pharmacology to choose the best drug while minimizing the number of tests.

Today, they tend to replace A/B testing for web site optimization (Google analytics), they are used for ad-serving optimization.
Make at simplest

When?
Partial information (multi classes problem)
just before the end

More Real-World Challenges for Data Stream Mining

Data stream research challenges positioned in the CRISP cycle.

"Open Challenges for Data Stream Mining Research", submitted to SIGKDD Explorations (Special Issue on Big Data)
Conclusion

Main ideas to retain:

• Online learning **algorithm** are designed in accordance with **specific constrains**
  – One pass
  – Low latency
  – Adaptive ... etc

• In practice the **true labels** are **delayed**: *an online classifier predicts the labels before observe it*

• The **evaluation** of the classifiers is **specific** to data streams processing

• The **distribution** of the tuples may **change over time**:
  – Some approaches **detect** the drifts, and then **update** the classifier (**abrupt drift**)
  – Other approaches **progressively adapt** the classifier (**incremental drift**)

• In practice, the type of **expected drift must be known** in order to choose an appropriate approach

• The distinction between **noise** and **drifts** can be viewed as a **plasticity / stability** dilemma