Data Stream Processing and Analytics

Vincent Lemaire

Thank to Alexis Bondu, EDF
Outline

• Introduction on data-streams
• Supervised Learning
• Conclusion
Big Data – what does that mean?

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

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 -> **Classifier**
- Numeric target variable -> **Regression**
- Time series -> **Forecasting**
A learning algorithm exploits the training set to automatically adjust the classifier.
From Batch mode to Online Learning

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.
FROM BATCH MODE TO ONLINE LEARNING

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
“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.

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

$X \rightarrow \text{Online Classifier} \rightarrow \hat{Y}$
Implementation of on-line classifiers

$X$ → Online Classifier → $\hat{Y}$ → Update → $Y$

Comparison of real and predicted labels

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

| Ad |

Online Classifier

P(Ad)

AgrMax(Ads)

Browser

Sending the Ad

Waiting for a click
Implementation of on-line classifiers

Example: online advertising targeting

User

Ad

Browser

Sending the Ad

Waiting for a click

Online Classifier

Update

P( hints )

Real labels

After a fixed delay

If clicked

$
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

Online Classifier

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

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>Classifier</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

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

Number
Granularity
Weights

Adaptation

Blind methods
- 'Informed methods'

It is necessary to adapt the classifier to the application context
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
  - ANN CAD (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

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>

Input stream: labeled examples
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?*
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
**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?

Incremental Decision Tree

The example of the Hoeffding Trees [D]
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 $\Delta G = \bar{G}(X_a) - \bar{G}(X_b) > \varepsilon$ after seeing $n$ examples at a node,
  - Hoeffding bound guarantees the true $\Delta G \geq \Delta \bar{G} - \varepsilon > 0$, 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**

- **Input stream**
  - Age < 30?
    - Car Type = Sports Car?
    - Status = Married?

  - Find the two best attributes
  - Check the condition \( G > \)

- **If not satisfied**
  - Create a new test at the current node
  - Split the stream of examples
  - Create 2 new leaves
  - Recursively run the algorithm on new leaves

- **If satisfied**
  - Car Type = Sports Car?
  - Status = Married?

**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 as a function of the number of training examples

Accuracy vs. # examples

Accuracy %

No. Examples

C4.5

VFDT

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
Incremental Decision Tree

Differents 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 entropie (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?
Incremental Decision Tree

Consider entropy $H(\rho)$

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

almost 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

\[ \text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log(p_1) - p_2 \log(p_2) \ldots - p_n \log(p_n) \]
Example: attribute “Outlook”

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

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

- Information gain:

\[(\text{information before split}) - (\text{information after split})\]

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

- Information gain for attributes from weather data:

\[
\text{gain("Outlook")} = 0.247 \text{ bits}
\]

\[
\text{gain("Temperature")} = 0.029 \text{ bits}
\]

\[
\text{gain("Humidity")} = 0.152 \text{ bits}
\]

\[
\text{gain("Windy")} = 0.048 \text{ bits}
\]
Continuing to split

gain("Temperature") = 0.571 bits

gain("Windy") = 0.020 bits

gain("Humidity") = 0.971 bits
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)
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

[Diagram of an incremental decision tree with a root node labeled 'age > 20', two children labeled 'yes' and 'no', each with a 'Leaf' and 'Summary' node, and two 'Local model' symbols.]
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
  ➔ naïve 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
Experimentations: Influence of the local model
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’
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
Concept drift

What does it mean?

- 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:

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

Original data

Virtual drift [B] (or covariate shift)

Concept drift [A]
Concept drift

What kinds of drift can be expected? [C]

- **Abrupt**
- **Gradual**
- **Incremental**
- **Reoccuring**

Drift detection

On-line adaptive learning

Drift detection & models management
Some specific constrains 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)*
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

Concept drift
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
adapting at every step

reactive, forgetting

Single classifier

ensemble
maintain some memory

Triggering
Evolving

Detectors
variable windows

Forgetting
fixed windows, Instance weighting

Contextual
dynamic integration, meta learning

Dynamic ensemble
adaptive combination rules

PAKDD-2011 Tutorial,
May 27, Shenzhen, China
A. Bifet, J. Gama, M. Pechenizkiy, I. Zliobaite
Handling Concept Drift: Importance, Challenges and Solutions
More details … see

Handling Concept Drift:
Importance, Challenges & Solutions

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

PAKDD-2011 Tutorial
May 27, Shenzhen, China

Concept drift

Drift detection

General schema:

- **Fixed Classifier (applied online)**

  - If detected:
    - Train a new classifier on the recent past
    - Adapt the size of the history

- **Drift Detection**

  - Replace the classifier
Drift detection

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
**Concept drift**

**Drift detection**

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 ...
Concept drift

Drift detection

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)
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)
Make at simplest

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
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