# Data Streaming with Affinity Propagation

Xiangliang Zhang, Cyril Furtlehner, and Michèle Sebag

TAO – INRIA CNRS
Université de Paris-Sud 11, F-91405 Orsay Cedex, France

**Abstract.** This paper proposed StrAP (Streaming AP), extending Affinity Propagation (AP) to data steaming. AP, a new clustering algorithm, extracts the data items, or exemplars, that best represent the dataset using a message passing method. Several steps are made to build StrAP. The first one (Weighted AP) extends AP to weighted items with no loss of generality. The second one (Hierarchical WAP) is concerned with reducing the quadratic AP complexity, by applying AP on data subsets and further applying Weighted AP on the exemplars extracted from all subsets. Finally StrAP extends Hierarchical WAP to deal with changes in the data distribution. Experiments on artificial datasets, on the Intrusion Detection benchmark (KDD99) and on a real-world problem, clustering the stream of jobs submitted to the EGEE grid system, provide a comparative validation of the approach.

## 1 Introduction

Data Streaming, one major task of Data Mining [1,2,3,4,5], aims to providing a compact description of the data flows generated by e.g., telecommunications, sensor networks, or Internet traffic. Data Streaming works under severe algorithmic constraints due to the size and dynamics of the data flow, since the underlying distribution of the data flow continuously evolves with the users, the usage and the application context.

Data Streaming is interested in various goals, e.g., computing approximate statistics [6,7], detecting novelties [1], or monitoring the top-K events [8] to name a few. This paper more specifically focuses on the identification of the clusters represented in the data stream, with the modelling of the jobs submitted to the EGEE grid[1] as motivating application.

The challenge is to achieve a good clustering behaviour, specifically enforcing a low distortion (more on this in Section 2) with low computational complexity, while swiftly adapting to the changes in the underlying distribution. An additional requirement is that a cluster should be represented by an actual data item (here, a job), as opposed to an average item, for the application domain hardly enables to consider artefacts. Under this requirement, clustering defines a combinatorial optimization problem, referred to as $K$-centers: i/ find the appropriate

[1] The EGEE grid, established in the EU project *Enabling Grid for e-Science in Europe*, `http://www.eu-egee.org/`) involves 41,000 CPUs and 5 Petabytes storage; it supports 20,000 concurrent jobs on a 24×7 basis.

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 628–643, 2008.

number $K$ of clusters; ii/ retain the best $K$ items, referred to as exemplars, which constitute the best representatives of all items.

The proposed approach is based on a new clustering algorithm accommodating the above requirement, called Affinity Propagation (AP) [9,10]. AP (Section 2.1) defines an energy-based formulation of the $K$-centers combinatorial optimization problem, which is solved using a message passing algorithm akin belief propagation. The direct use of AP for Data Streaming however raises two difficulties. Firstly, AP has been designed for batch clustering, and it must be adapted to online clustering to handle data streams. Secondly, and most importantly, AP suffers from its quadratic computational complexity in the number $N$ of items.

The work presented in this paper extends AP to Data Streaming, and proposes three contributions. The first one extends AP to deal with duplicated items with no loss of performance (Weighted AP algorithm, WAP). The second one, called Hierarchical WAP (Hi-WAP), aims at decreasing the computational complexity to $\mathcal{O}(N^{1+\alpha})$ ($\alpha > 0$), taking inspiration from [11]: the idea is to partition the dataset, run AP on each subset, and apply WAP to the collection of exemplars constructed from each subset. The third one, StrAP, achieves online clustering, storing the outliers and occasionally updating the exemplars. Two update criteria have been considered. The first one is based on the number of outliers; when it reaches a predefined threshold, Hi-WAP is applied to the current exemplars and the outliers. The second one is based on the so-called Page-Hinkley change-point detection statistical test [12,13], observing the outlier rate. When a change point is detected, Hi-WAP is likewise applied on the current exemplars and the outliers. The experimental validation on artificial datasets, on the KDD99 benchmark dataset, and on the real-world dataset of the EGEE jobs, demonstrates the relevance of the approach compared to $K$-centers and the *DenStream* algorithm [4].

The paper is organized as follows. After presenting AP for the sake of completeness, Section 2 describes the first two extensions, Weighted and Hierarchical AP. Section 3 gives an overview of StrAP. Section 4 describes the experimental setting and the datasets used for the experimental validation of the approach. Finally, Section 5 and 6 report on the experimental results, using K-centers and *DenStream* as baselines. The approach is discussed w.r.t. related work and the paper concludes with some perspectives for further research.

## 2 Affinity Propagation and Scalable Extensions

For the sake of self-containedness, this section first describes the AP algorithm, referring the reader to [9,10] for a comprehensive introduction. Two AP extensions are thereafter described, respectively handling the case of weighted items, and the merge of partial solutions.

### 2.1 Affinity Propagation

Let $\mathcal{E} = \{e_1, \ldots e_N\}$ be a set of items, and let $d(i,j)$ denote the distance or dissimilarity between items $e_i$ and $e_j$. Letting $K$ denote a positive integer, the

$K$-center problem consists of finding $K$ items in $\mathcal{E}$, referred to as exemplars and denoted $e_{i_1}, \ldots, e_{i_K}$, such that they minimize the sum, over all items $e_j$, of the minimal squared distance between $e_j$ and $e_{i_k}, k = 1 \ldots K$.

The Affinity Propagation approach proposes an equivalent formalization of the $K$-center problem, defined in terms of energy minimization. Let $\sigma(i)$ associate to each item $e_i$ the index of its nearest exemplar, then the goal is to find the mapping $\sigma$ maximizing the functional $E[\sigma]$ defined as:

$$E[\sigma] = \sum_{i=1}^{N} S(e_i, e_{\sigma(i)}) - \sum_{i=1}^{N} \chi_i[\sigma] \tag{1}$$

where $S(e_i, e_j)$ is set to $-d(i,j)^2$ if $i \neq j$, and is set to a small constant $-s^*$, $s^* \geq 0$ called *preference* otherwise. The second term in the energy function expresses that if $e_i$ is selected as an exemplar by some items, it has to be its own exemplar[2], with $\chi_i[\sigma] = \infty$ if $\sigma(\sigma(i)) \neq \sigma(i)$ and 0 otherwise.

Aside from the consistency constraints, the energy function thus enforces a tradeoff between the distortion, i.e. the sum over all items of the squared error $d(i, \sigma(i))^2$ committed by assimilating item $e_i$ to its nearest exemplar $e_{\sigma(i)}$, and the cost of the model, that is $s^* \times |\sigma|$ if $|\sigma|$ denotes the number of exemplars retained. Eq. (1) thus does not directly specify the number of exemplars to be found, as opposed to $K$-centers. Instead, it specifies the penalty $s^*$ for allowing an item to become an exemplar; note that for $s^* = 0$, the best solution is the trivial one, selecting every item as an exemplar.

The resolution of the optimization problem defined by Eq. (1) is achieved by a message passing algorithm, considering two types of messages: availability messages $a(i,k)$ express the accumulated evidence for $e_k$ to be selected as the best exemplar for $e_i$; responsibility messages $r(i,k)$ express the fact that $e_k$ is suitable to be the exemplar of $e_i$.

All availability and responsibility messages $a(i,k)$ and $r(i,k)$ are set to 0 initially. Their values are iteratively adjusted[3] by setting:

$$\begin{aligned}
r(i,k) &= S(e_i, e_k) - \max_{k', k' \neq k}\{a(i,k') + S(e_i, e'_k)\} \\
r(k,k) &= S(e_k, e_k) - \max_{k', k' \neq k}\{S(e_k, e'_k)\} \\
a(i,k) &= min\left\{0, r(k,k) + \textstyle\sum_{i', i' \neq i,k} \max\{0, r(i',k)\}\right\} \\
a(k,k) &= \textstyle\sum_{i', i' \neq k} \max\{0, r(i',k)\}
\end{aligned}$$

The index of exemplar $\sigma(e_i)$ associated to $e_i$ is finally defined as:

$$\sigma(i) = argmax\left\{r(i,k) + a(i,k), k = 1 \ldots N\right\} \tag{2}$$

The algorithm is stopped after a maximal number of iterations or when the exemplars did not change for a given number of iterations.

[2] This constraint is relaxed by the soft-constraint AP (SCAP) [14], unveiling the hierarchical cluster structure in the data set while AP is biased toward regularly shaped clusters. The extension of the presented approach to SCAP is left for further study.

[3] Numerical oscillations are avoided by using a relaxation mechanism; empirically, the actual value is set to the half sum of the old and new values [9].

As could have been expected, Affinity Propagation is not to be seen as a universally efficient data clustering approach. Firstly, linear and robust algorithms such as $K$-means should be preferred to AP in domains where artefact items can be constructed[4]. Secondly, and most importantly, AP suffers from a quadratic computational complexity in the number $N$ of items: on the one hand, dissimilarities $d(i,j)$ must be computed; on the other hand, the message passing algorithm converges with complexity $\mathcal{O}(N^2 logN)$, hindering its direct use in large-scale applications. By convention, in the following notation $\widetilde{\mathcal{O}}(P)$ will stand for $\mathcal{O}(P \times polynom(\log P))$.

### 2.2 Weighted and Hierarchical WAP

Two extensions of AP, aiming at a lesser computational complexity, are presented in this section.

**Weighted AP.** A preliminary step is to extend AP in order to deal with multiply-defined items. Let dataset $\mathcal{E}' = \{(e_i, n_i)\}$ involve $n_i$ copies of item $e_i$, for $i = 1 \dots L$. Let us consider the dissimilarity matrix $S'$ defined as:

$$S'(e_i, e_j) = \begin{cases} -n_i d(i,j)^2 & \text{if } i \neq j \\ s^* + (n_i - 1) \times \varepsilon_i & \text{otherwise}, \varepsilon_i \geq 0 \end{cases}$$

**Proposition.** The combinatorial optimization problem of finding $\sigma : \{1 \dots L\}$ minimizing

$$E'[\sigma] = \sum_{i=1}^{L} S'(e_i, e_{\sigma(i)}) - \sum_{i=1}^{L} \chi_i[\sigma] \tag{3}$$

is equivalent, for $\varepsilon_i = 0$, to the optimization problem defined by Eq. (1) for $\mathcal{E}$ made of the union of $n_i$ copies of $e_i$, for $i = 1 \dots L$.

*Proof*
*In the optimization problem defined by Eq. (1), assume that $e_i$ actually represents a set of $n_i$ identical copies; the penalty $S(e_i, e_j)$ of selecting $e_j$ as exemplar of $e_i$ thus is the cost of selecting $e_j$ as exemplar for each one of these copies. Therefore $S'(e_i, e_j) = n_i \times (-d(i,j)^2)$.*

*Likewise, let $e_i$ be unfolded as a set of $n_i$ (almost) identical copies $\{e_{i_1}, \dots, e_{i_{n_i}}\}$, and let us assume that one of them, say $e_{i_1}$ is selected as exemplar. One thus pays the* preference *penalty $s^*$, plus the sum of the dissimilarities between $e_{i_1}$ and the other copies in $e_i$, modelled as $(n_i - 1)\varepsilon_i$. Constant $\varepsilon_i$ thus models the average dissimilarity among the $n_i$ copies of $e_i$.*

**Hierarchical WAP.** Hierarchical WAP proceeds by launching AP on subsets of the current dataset, and thereafter launching WAP on the dataset made of all exemplars extracted from the subsets.

[4] Selecting the best set of artefacts out of $\tau$ independent runs of $K$-means usually enforce a high-quality distortion, with complexity $\tau \times K \times N$.

Formally, let dataset $\mathcal{E}$ be equally divided into $\sqrt{N}$ subsets noted $\mathcal{E}_i, i = 1 \ldots \sqrt{N}$. Running AP on $\mathcal{E}_i$ outputs $K_i$ exemplars noted $\{e_{i_1}, \ldots e_{i_{K_i}}\}$; let us denote $n_{i_j}$ the number of items in $\mathcal{E}_i$ having $e_{i_j}$ as nearest exemplar.

Define $\mathcal{E}' = \{(e_{i_j}, n_{i_j}), i = 1 \ldots \sqrt{N}, j = 1 \ldots K_i\}$. Hi-WAP launches WAP on $\mathcal{E}'$, and returns the produced exemplars.

**Proposition.** The complexity of Hi-WAP is $\widetilde{\mathcal{O}}(N^{\frac{3}{2}})$.
*Proof. The construction of $\mathcal{E}'$ is in $\widetilde{\mathcal{O}}(N^{\frac{3}{2}})$, since AP is applied $\sqrt{N}$ times on datasets of size $\sqrt{N}$.*

*Letting $K$ be an upper bound on the number of exemplars learned from every subset $\mathcal{E}_i$, WAP thus achieves the distributed clustering of the exemplars extracted from all $\mathcal{E}_i$ with complexity $\widetilde{\mathcal{O}}(N^{\frac{1}{2}} \times K^2)$. The total complexity then is $\widetilde{\mathcal{O}}(NK^2 + N^{\frac{3}{2}})$, where term $N^{\frac{3}{2}}$ is dominant since $\sqrt{N} > K$.*

Iterating this hierarchical decomposition along the same lines as [11] leads to decrease the complexity to $\widetilde{\mathcal{O}}(N^{1+\alpha})$; on-going research is concerned with bounding the loss of distortion incurred by Hi-WAP. Experimentally, the distortion loss is found to be very moderate while the computational cost decreases by one order of magnitude or more. Therefore, in the following we will use indifferently AP or Hi-WAP, referred to as *AP, depending on the pressure on the computational resources and the size of the data.

## 3 Data Streaming with AP

This section describes the StrAP algorithm, extending AP to Data Streaming, involving four main steps (Alg. 1):

1. The first bunch of data is used by *AP to compute the first exemplars and initialize the stream model.
2. As the stream flows in, each data item $e_t$ is compared to the exemplars; if too far from the nearest exemplar, $e_t$ is put in the reservoir, otherwise the stream model is updated accordingly (section 3.1).
3. The restart criterion is triggered if the number of outliers exceeds the reservoir size, or upon a change point detection in the data distribution (section 3.2).
4. If it is triggered, the stream model is rebuilt from the current exemplars and the reservoir, using *AP again (section 3.3).

The stream model is available at any time. The performance of the process is measured from the average distortion and the clustering accuracy (section 3.4).

### 3.1 AP-Based Model and Update

The model of the data stream used in StrAP is inspired from *DbScan* [15] and *DenStream* [4]. It consists of 4-tuple $(e_i, n_i, \Sigma_i, t_i)$, where $e_i$ ranges over the exemplars, $n_i$ is the number of items associated to exemplar $e_i$, $\Sigma_i$ is the distortion of $e_i$ (sum of $d(e, e_i)^2$, where $e$ ranges over all items associated to $e_i$), and $t_i$ is the last time stamp when an item was associated to $e_i$.

For each new item $e_t$, its nearest exemplar $e_i$ is computed; if $d(e_t, e_i)$ is less than some threshold $\varepsilon$, heuristically set to the average distance between points and exemplars in the initial model, $e_t$ is affected to the $i$-th cluster and the model is updated accordingly; otherwise, $e_t$ is considered to be an outlier, and put in the reservoir.

In order to avoid the number of exemplars to grow beyond control, one must be able to forget the exemplars that have not been visited for some time. Accordingly, a used-specified window length $\Delta$ is considered; when item $e_t$ is associated to exemplar $e_i$, the model update is thus defined as:

$$n_i := n_i \times \left(\frac{\Delta}{\Delta+(t-t_i)} + \frac{1}{n_i+1}\right) \qquad \Sigma_i := \Sigma_i \times \frac{\Delta}{\Delta+(t-t_i)} + \frac{n_i}{n_i+1} d(e_t, e_i)^2 \qquad t_i := t$$

Simple calculations show that the above update rules enforce the model stability if exemplar $e_i$ is selected on average by $n_i$ examples during the last $\Delta$ time steps. The sensitivity analysis w.r.t. $\Delta$ is discussed in section 6.2.

### 3.2 Restart Criterion

A key difficulty in Data Streaming is to tell an acceptable ratio of outliers from a change in the generative process underlying the data stream, referred to as drift. In case of drift, the stream model must be updated. In some domains, e.g., continuous spaces, smooth updates can be achieved by gradually moving the centers of the clusters. When artefacts cannot be considered, the centers of the clusters must be redefined. In such domains, the data streaming process thus needs a restart criterion, in order to decide whether to launch the selection of new exemplars.

Two restart criteria have been considered. The first one is most simply based on the number of outliers in the reservoir; when it exceeds the reservoir size, the restart criterion is triggered. The second criterion is based on the distribution of the data items[5]. Let us consider the sequence of items $e_t$; define $p_t$ as $c/(1+o_t)$ where $o_t$ is the fraction of non-outliers and $c$ is 1 (resp. 2) if $e_t$ is an outlier (or not). If a drift occurs in the data distribution, then sequence $p_t$ should display some change; the restart criterion is triggered upon detecting such a change.

Among the many change point detection tests, the so-called Page-Hinkley test (PH) [12,13] has been selected as it minimizes the expected detection time for a prescribed false alarm rate. Formally, the PH test is controlled after a detection threshold $\lambda$ and tolerance $\delta$, as follows:

$$\begin{array}{ll} \bar{p}_t = \frac{1}{t}\sum_{\ell=1}^{t} p_\ell & m_t = \sum_{\ell=1}^{t} (p_\ell - \bar{p}_\ell + \delta) \\ M_t = max\{m_\ell, \ell = 1...t\} & PH_t = (M_t - m_t) > \lambda \end{array}$$

Parameter $\delta$ is set to $10^{-2}$ in all experiments. The sensitivity analysis w.r.t. $\lambda$ is presented in section 6.1.

[5] In case the number of outliers exceeds reservoir size, the new outlier replaces the oldest one in reservoir; a counter keeping track of the removed outliers is incremented.

### 3.3 Model Rebuild

Upon triggering of the restart criterion, Weighted AP is launched on $\mathcal{E} = \{(e_i, n_i)\} \cup \{(e'_j, 1)\}$, where $(e_i, n_i)$ denotes an exemplar of the current stream model together with the associated size $n_i$, and $e'_j$ is an outlier item in the reservoir. Penalties are defined after section 2.2, as follows:

$$\begin{array}{ll} S(e_i, e_i) = s^* + \Sigma_i & S(e'_j, e'_j) = s^* \\ S(e_i, e_j) = -n_i\, d(e_i, e_j)^2 & S(e_i, e'_j) = -n_i\, d(e_i, e'_j)^2 \\ S(e'_j, e_i) = -d(e_i, e'_j)^2 & \end{array}$$

After the new exemplars have been selected by WAP from $\mathcal{E}$, the stream model is defined as follows. Formally, let $f$ denote a new exemplar and let $e_1, \ldots e_m$ (respectively $e'_1, \ldots, e'_{m'}$) be the previous exemplars (resp. reservoir items) associated to $f$. With no difficulty, the number $n$ of items associated to $f$ is set to $n_1 + \ldots + n_m + m'$. The associated distortion $\Sigma$ is estimated as follows. Let $e$ be an item associated to $e_1$. Indeed $e$ is no longer available; but assuming an Euclidean space, $e$ can be modelled as a random item $e_1 + X\boldsymbol{v}$, where $\boldsymbol{v}$ is a random vector in the unit ball, and $X$ is a scalar random variable with normal distribution. It comes:

$$\begin{aligned} ||f - e||^2 &= ||f - e_1||^2 + ||e_1 - e||^2 - 2\langle f - e_1, X\boldsymbol{v}\rangle \\ &= d(f, e_1)^2 + d(e_1, e)^2 - 2X\langle f - e_1, \boldsymbol{v}\rangle \end{aligned}$$

Therefore, taking the expectation, $\mathbb{E}[d(f, e)^2] = d(f, e_1)^2 + \frac{1}{n_1}\Sigma_1$. Accordingly,

$$\Sigma = \sum_{i=1}^{m} \left(n_i d(f, e_i)^2 + \Sigma_i\right) + \sum_{i=1}^{m'} d(f, e'_i)^2$$

Finally, $t$ is set to the maximal time stamp associated to $e_i$ and $e'_j$, for $e_i$ and $e'_j$ ranging among the exemplars and outliers associated to $f$.

**Algorithm 1.** StrAP Algorithm

```
Datastream e_1, ... e_t, ...; fit threshold ε
Init
   *AP(e_1, ..., e_T) → StrAP Model                    section 3.1
   Reservoir = {}
for t > T do
   Compute e_i = nearest exemplar to e_t               section 3.1
   if d(e_t, e_i) < ε then
      Update StrAP model                               section 3.1
   else
      Reservoir ← e_t
   end if
   if Restart criterion then                           section 3.2
      Rebuild StrAP model                              section 3.3
      Reservoir = {}
   end if
end for
```

### 3.4 Evaluation Criterion

**Distortion.** The performance of StrAP is first measured after the overall distortion $D$, measured as follows. When a new item $e_t$ is associated to exemplar $e_i$, $D$ is incremented by $d(e_t, e_i)^2$. The distortion due to outliers is estimated a posteriori; after every restart, the average square distance $\bar{d}^2$ of the reservoir items to the new exemplars is computed, and $D$ is incremented by $\bar{d}^2$ times the number of items put in the reservoir since the previous restart (taking into account the outliers removed from reservoir when using PH restart criterion, section 3.2).

**Accuracy of Clustering.** In the case where the items are labeled, the clustering quality is also commonly assessed after the purity of the clusters. An item associated to an exemplar is correctly classified (respectively, misclassified) if its class is same (resp. different) as the exemplar class. The accuracy, the error rate and the percentage of outliers, sum up to 100%.

## 4 Goal of the Experiments and Setting

The algorithms presented in sections 2.2 and 3 raise two main questions.

The first question is whether Hi-WAP, designed for reducing the computational effort of AP, does so at the expense of a significant increase of the distortion. The tradeoff between the computational cost and the distortion will be assessed by experimentally comparing Hi-WAP with (batch) AP, on the one hand, and with other hierarchical variants (involving $K$-centers and AP, as opposed to WAP) on the other hand (Section 5). The experiments firstly involve benchmark datasets, kindly provided by E. Keogh [16]. As the focus essentially concerns the scalability of Hi-WAP, only the largest two datasets (Faces and Swedish leaves, respectively including 2250 and 1125 examples) have been considered. Secondly, a real-world dataset describing the 237,087 jobs submitted to the EGEE grid system, has been considered. Each job is described by five attributes:

1. the duration of waiting time in a queue;
2. the duration of execution;
3. the number of jobs waiting in the queue when the current job arrived;
4. the number of jobs being executed after transiting from this queue;
5. the identifier of queue by which the job was transited.

Note that the behavior might be significantly different from one queue to another. The expert is willing to extract representative actual jobs (as opposed to virtual ones, e.g. executed on queue 1 with weight .3 and on queue 2 with weight .7), which is the main applicative motivation for using AP. The dissimilarity of two jobs $x_i$ and $x_j$ is the sum of the Euclidean distance between the numerical description of $x_i$ and $x_j$, plus a weight $w_q$ if $x_i$ and $x_j$ are not executed on the same queue. Further, the EGEE dataset involves circa 30% duplicated items (different jobs with same description).

The second question regards the performance of StrAP algorithm. As a clustering algorithm, it is meant to enforce a low distortion and high purity; as a streaming algorithm, it is required to efficiently adapt to the changing distribution of data stream. StrAP performances are assessed comparatively to those of *DenStream* [4] (Section 6), using an artificial stream generator, and the Intrusion Detection benchmark data set referred to as KDD Cup 1999 [17,18].

The stream generator is parameterized from the dimension $D$ of the data items, and the number $M$ of target exemplars. Each target exemplar $e_i$ is uniformly selected in $[-1, 1]^D$ and its probability $p_i(t)$ evolves along time proportionally to $w_i \times sin(\omega_i t + \varphi_i)$, where weight $w_i$, frequency $\omega_i$ and phase $\varphi_i$ are uniformly selected respectively in $[1, 100]$, $[0, 2\pi]$, and $[-\pi/2, \pi/2]$. At time step $t$, exemplar $e_i$ is selected with probability $p_i(t)$, and item $e_t$ is set to $e_i$ plus gaussian noise.

The Intrusion Detection dataset we used includes 494,021 network connection records (71MB). Records are distributed among 23 classes, the *normal* class and the specific kinds of attack, such as *buffer_overflow*, *ftp_write*, *guess_passwd*, *neptune*. Out of the 41 attributes, only the numeric 34 ones have been used after [4], and cast such as they have same range of variation[6].

All reported computational times have been measured on Intel 2.66GHz Dual-Core PC with 2 GB memory.

## 5 Experimental Validation of Hi-WAP

This section reports on the performances of Hi-WAP, specifically the tradeoff between the computational effort and the distortion, on benchmark datasets and the real-world EGEE dataset.

### 5.1 Experimental Setting

On each dataset $\mathcal{E}$ of size $N$, the experiments were conducted as follows:

- $\mathcal{E}$ is partitioned into $\sqrt{N}$ subsets of equal size noted $\mathcal{E}_i$.
- Hi-WAP (respectively Hi-AP):
  1. On $\mathcal{E}_i$, the preference $s_i^*$ is set to the median of the pair similarities in the subset. WAP (respectively AP) is launched and produces a set of exemplars.
  2. WAP (respectively AP) is launched on the union of the exemplar set, varying preference $s^*$ from the minimum to the median distance of the exemplar pairs.
- Hierarchical $K$-centers:
  1. In parallel, $K$-centers is launched 120 times on each $\mathcal{E}_i$, where $K$ is set to the average number of exemplars extracted from the $\mathcal{E}_i$. The best set of exemplars (w.r.t. distortion) is retained; let $\mathcal{C}$ denote the union of these best sets of exemplars.

[6] Attribute *duration* is changed from seconds into minutes; *src_bytes* and *dst_bytes* are converted from byte to KB; *log* is used on *count*, *srv_count*, *dst_host_count*, *dst_host_srv_count*.

2. For each $K$, varying in the interval defined by the number of clusters obtained by Hi-WAP, 20 independent runs of $K$-centers are launched on $\mathcal{C}$, and the best set of exemplars is returned. The number of independent runs is such that Hierarchical $K$-centers and Hi-WAP have same computational cost for a fair comparison.

– The two approaches are graphically compared, reporting the distortion vs the number of clusters obtained by respectively Hierarchical $K$-Centers, Hi-WAPand Hi-AP.

### 5.2 Experimentation on Benchmark Dataset

As mentioned in Section 4, the benchmark datasets involve the largest two datasets from E. Keogh [16]. Table 1 displays the distortion obtained by $K$-centers and AP in non hierarchical and hierarchical settings. The number $K$ of clusters is set to the number of classes (Table 1.(a)) or defined by AP (Table 1.(b); $s^*$ is set to the median distance). $N$ and $D$ respectively stand for the number of items and the number of dimensions of the dataset.

After Table 1.(a), the loss of distortion incurred by Hi-WAP compared with AP is about 3.3% in the face dataset, and 5.7% in the Swedish leaf case; the distortion is about the same as for (non hierarchical) $K$-centers. The computational time is not relevant here due to the extra-cost of setting $s^*$ in order to enforce the desired number of clusters.

After Table 1.(b), AP significantly improves on $K$-centers (left part) in terms of distortion, by 14% in the face dataset and 27% in the Swedish leaf dataset; Hi-WAP similarly improves on hierarchical $K$-centers (right part), by 7% in the face dataset and 2% in the Swedish leaf dataset. Hierarchical variants improve over batch one by at least one order of magnitude in terms of computational

**Table 1.** Experimental results: Comparative Distortion of $K$-centers, AP, Hierarchical $K$-centers, Hi-AP and Hi-WAP. All results reported for $K$-centers variants are the best ones obtained with same computational effort as for AP variants.

(a). The number $K$ of clusters is set to the number of classes.

| Data | K | N | D | Non Hierarchical | | Hierarchical | | |
|---|---|---|---|---|---|---|---|---|
| | | | | $KC$ | AP | $KC$ | Hi-AP | Hi-WAP |
| Face (all) | 14 | 2250 | 131 | 189370 | **183265** | 198658 | 190496 | **189383** |
| Swedish Leaf | 15 | 1125 | 128 | 20220 | **19079** | 20731 | 20248 | **20181** |

(b). $K$ is fixed after AP or Hi-WAP, for $s^*$ set to the median distance.

| Data | N | D | Non Hierarchical | | | Hierarchical | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | K(AP) | $KC$ | AP | K(Hi-WAP) | $KC$ | Hi-AP | Hi-WAP |
| Face (all) | 2250 | 131 | 168 | 100420 (128 sec) | **88282** | 39 | 172359 (3 sec) | 164175 | **160415** |
| Swedish Leaf | 1125 | 128 | 100 | 12682 (21 sec) | **9965** | 23 | 21525 (1.4 sec) | **20992** | 21077 |

cost; the distortions cannot be directly compared as the number of clusters is different.

### 5.3 Experimentation on the EGEE Dataset

The real-world dataset describing the jobs submitted to the EGEE grid (section 4) is used to compare the distortion incurred by hierarchical clusterings, $K$-centers, Hi-AP and Hi-WAP, after the same procedure as in section 5.1; AP cannot be used on this dataset as it does not scale up. The number $K$ of clusters for $K$-centers is set to 15; 120 independent $K$-centers runs are launched, and the best distortion is reported, for a fair comparison (same computational cost and overall number of clusters). The first phase (clustering all $\sqrt{N}$ datasets) amounts to 10 minutes for $K$-centers and Hi-WAP, and 26 minutes for Hi-AP due to the duplications in the dataset). Note that the computational cost of this first phase can trivially be decreased by parallelization.

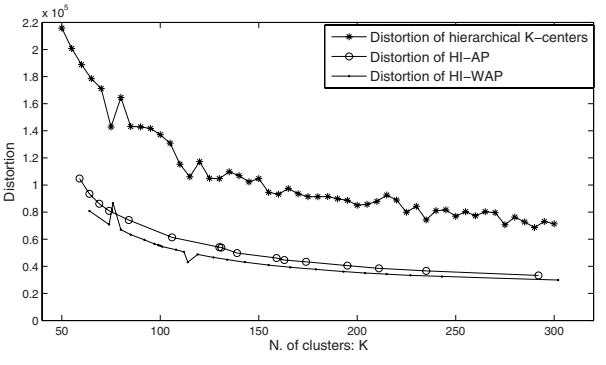



**Fig. 1.** Distortion of hierarchical $K$-centers, Hi-AP and Hi-WAP on the EGEE job dataset

The various distortions obtained when varying the number $K$ of clusters and the preference $s^*$ are reported in Fig. 1, showing that Hi-WAP improves on both Hi-AP and $K$-centers; the distortion is decreased by a factor 2 compared to $K$-centers and the computational cost (not shown) is decreased by a factor 3 compared to Hi-AP.

## 6 Experimental Validation of StrAP

This section reports on the experimental validation of StrAP on a synthetic dataset and on the Intrusion Detection dataset (KDD99), described in Section 4. Results obtained on the EGEE dataset are omitted due to lack of space.

The initialization of the stream model (Section 3) considers the first 800 (synthetic data) or 1000 (KDD99) items.

### 6.1 Synthetic Data Stream

The dynamics of the synthetic data stream is depicted on Fig. 2; the only clusters represented at the beginning are clusters 2, 7, and 9. Representatives of cluster 0 appear shortly after the initialization; they are first considered to be outliers (legend $*$); using the Page-Hinkley restart criterion ($\lambda$=5, $\delta$=0.01), the first restart indicated by a vertical line occurs soon after. The same pattern is observed when the first representatives of clusters 3, 5 and 8 appear; they are first considered to be outliers, and they respectively trigger the second, third and fourth restarts thereafter. The small number of the "true" clusters makes it unnecessary to use a windowing mechanism (section 3.1).

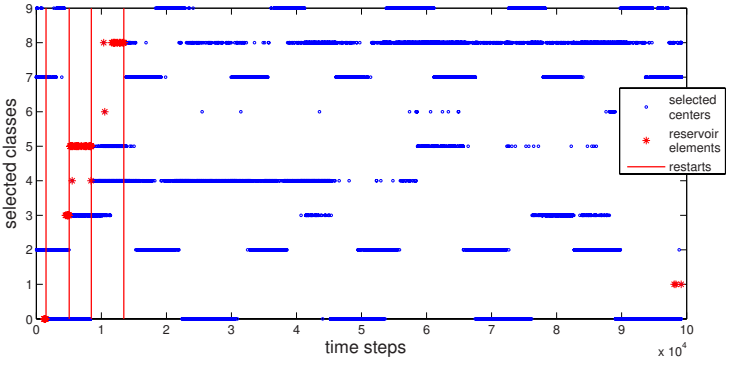



**Fig. 2.** Performance of StrAP on the synthetic data stream

Table 2 displays the performances of StrAP with respect to the percentage of outliers, the error rate[7], and the distortion, depending on the restart criterion used, and the parameters thereof.

**Table 2.** Experimental results of StrAP on the synthetic data stream

| Restart | | Outlier (%) | Error | N. of restart | N. of clusters | Distortion | Runtime |
|---|---|---|---|---|---|---|---|
| | 5 | 0.13 | 0 | 4 | 16 | 4369521 | |
| PH, | 10 | 0.17 | 0 | 4 | 16 | 4369410 | 19 sec |
| $\delta$=0.01, $\lambda$= | 20 | 0.62 | 0 | 4 | 20 | 4159085 | |
| Maximum | 50 | 0.20 | 0 | 4 | 16 | 4334710 | |
| size of | 100 | 0.41 | 0 | 4 | 19 | 4116768 | 20 sec |
| reservoir | 300 | 1.34 | 0 | 4 | 25 | 3896671 | |

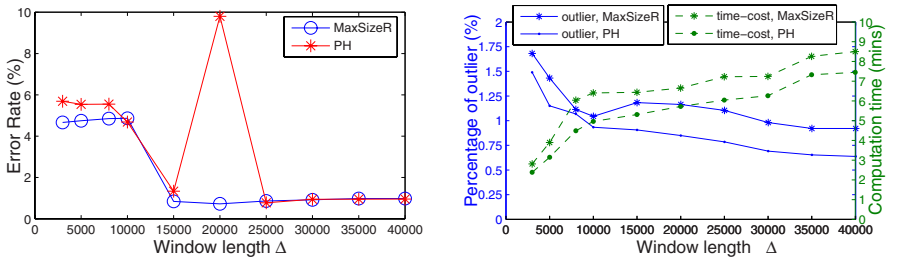
All clusters are pure. Interestingly, when the restart criterion becomes less sensitive (increasing parameter $\lambda$ or reservoir size $MaxSizeR$), the outlier rate

[7] The classes of the items in the synthetic dataset correspond to the original exemplars respectively used to generate the items.

increases together with the number of clusters, while the number of restarts remains constant. A tentative interpretation is that, the less sensitive the restart, the more outliers and the more diverse the reservoir becomes; this diversity results in a higher number of clusters, decreasing the distortion. The computational time is circa 20 seconds for 100,000 items. Similar results are obtained for various number of dimensions ($D = 30$ in Table 2).

### 6.2 Intrusion Detection Dataset

The 494,021-transaction Intrusion Detection dataset is handled as a stream after [4]. StrAP performances are measured and compared with those of *Denstream*, in terms of error rate, outlier rate, and computational time. The sensitivity of results w.r.t. window length parameter $\Delta$ (section 3.1) is reported on Fig. 3.




(a) Clustering accuracy (b) Percentage of outlier, computation time

**Fig. 3.** Performance of StrAP on KDD99 dataset: comparing restart criterion PH ($\lambda = 20$) and Reservoir size ($MaxSizeR = 300$), depending on window length $\Delta$

Fig. 3 (a) shows that StrAP clustered the Intrusion Detection data stream with very low error rate (less than 1% for $\Delta > 15000$; the error peak observed for $\Delta = 20,000$ with the Page-Hinkley criterion is being investigated).

Fig. 3.(b) shows that the PH criterion improves on the Reservoir size, with a lower percentage of outliers and a smaller computational time, though the difference is not significant. The runtime is circa 7 minutes. It is worth noting that StrAP only needs 1% of the data (initial subset plus the outliers) in order to produce an accurate model (less than 1% error rate).

The online performance of StrAP is displayed in Fig. 4, reporting the error rate along time for $\Delta = 15000$ and $MaxSizeR = 300$; restarts are indicated with stars.

Fig. 5 presents a comparative assessment of StrAP and *DenStream* [4], using the same purity measure:

$$Purity = 100 \times (\sum_{i=1}^{K} \frac{|C_i^d|}{|C_i|})/K$$

where $K$ is the number of clusters, $|C_i|$ is the size of cluster $i$ and $|C_i^d|$ is the number of majority class items in cluster $i$. The clustering purity of *DenStream*

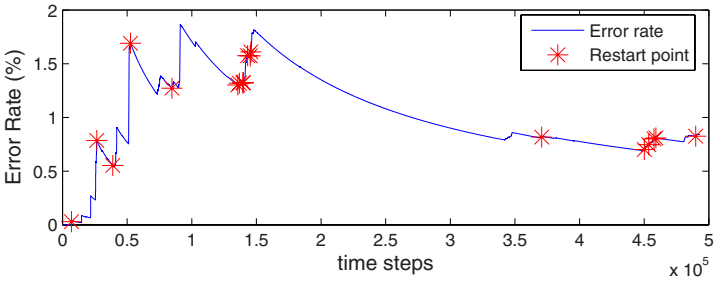



**Fig. 4.** Accuracy of StrAP on KDD99 data when $\Delta = 15000$ and $MaxSizeR = 300$

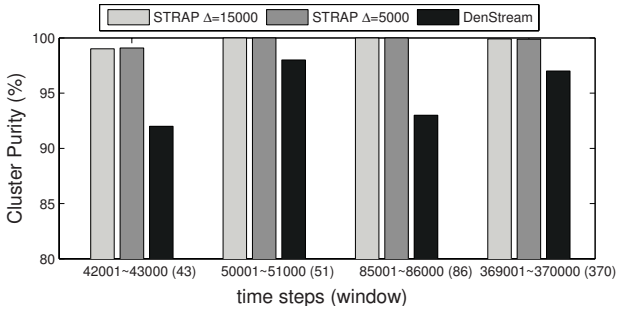



**Fig. 5.** Comparative performances of StrAP and *DenStream* on the Intrusion Detection dataset

on the Intrusion Detection dataset was evaluated during four time windows of length 1000 when some attacks happened. For a fair comparison, the clustering purity of StrAP was computed during the same time windows, considering the same 23 classes.

Fig. 5 respectively reports the results obtained for StrAP with one of the best settings ($\Delta = 15000$ and $MaxSizeR = 300$), an average setting ($\Delta = 5000$ and $MaxSizeR = 300$), and the results of *DenStream* found in [4]. In both StrAP settings, similar number of clusters are obtained (respectively circa 45, 32, 55 and 8 in the four windows).

On the Intrusion dataset, StrAP thus consistently improves on *DenStream*; as a counterpart, the computational time is higher by one or two orders of magnitude ( 7 minutes against 7 seconds for *DenStream*), noting that StrAP is written in Matlab.

### 6.3 Discussion

While many data streaming algorithms actually focus on the extraction of statistical information from data streams [6,7,8], ranging from the approximation of frequent patterns [19] to the construction of decision trees [20], the most related work is that of [4], similarly addressing unsupervised learning and clustering from data streams. The *DenStream* algorithm upgrades the *DbScan* clustering

algorithm [15] to dynamic environments; it mainly differs from StrAP regarding the creation and update of clusters. Actually, *DenStream* does not construct the final clusters unless requested to do so by the user; upon such a request, the (most recent) items will be labeled after the clusters. While this "lazy" clustering and labeling behavior is more computationally efficient, it is suggested that it is not well-suited to e.g., monitoring applications, when the goal is to identify behavioral drifts as soon as they appear.

Another relevant work, presented by Cormode et al. [21], aims at the structure of clusters in the stream. Interestingly, an extension of AP referred to as Soft-Constraint AP (SCAP) has been proposed by [14]; SCAP is concerned with identifying the relations between the exemplars. Further research will investigate the extension of SCAP to data streaming, to address the structured cluster extraction from data streams. Further work will also consider the detection of anomalies (see e.g. [22]), considering outliers (or small clusters) as candidate anomalies, and using StrAP as a pre-filter for anomaly detection.

## 7 Conclusion

The main contribution of this paper is to extend the Affinity Propagation algorithm proposed by Frey and Dueck [9] along two perspectives. The first one concerns the computational scalability; the ability to deal with large datasets is indeed becoming a crucial requirement in Machine Learning and Data Mining. The second one, concerned with online clustering, aimed at dealing with evolving data distributions, and seamlessly updating the data model.

These extensions, encapsulated in the StrAP algorithm, have been empirically validated in the data streaming context, on two large sized datasets including the Intrusion Detection dataset used as KDD99 benchmark problem, and compared to the state-of-the art *DenStream* algorithm. While the accuracy of the data model constructed by StrAP was found comparatively satisfactory, the computational time is higher than for *DenStream*.

A first priority for further study is to provide theoretical guarantees for the Hi-WAP algorithm; while it has been shown that the divisive schema can be used to cut down the computational complexity and bring it down to a super-linear one, it is most desirable to provide theoretical guarantees on the loss of distortion incurred along the divisive process. The second priority regards a main limitation of AP, namely the fact that the number of clusters is only indirectly controlled from the preference parameter $s^*$. A strategy for adjusting this parameter, either globally after the desired number of clusters, or locally (depending on the estimated density of the dataset), would alleviate this limitation.

## References

1. Fan, W., Wang, H., Yu, P.: Active mining of data streams. In: SIAM Conference on Data Mining (SDM) (2004)
2. Aggarwal, C., Han, J., Wang, J., Yu, P.: A framework for clustering evolving data streams. In: Int. Conf. on Very Large Data Bases(VLDB), pp. 81–92 (2003)

3. Guha, S., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams. In: IEEE Symposium on Foundations of Computer Science, pp. 359–366 (2000)
4. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SIAM Conference on Data Mining (SDM) (2006)
5. Muthukrishnan, S.: Data streams: Algorithms and applications. Found. Trends Theor. Comput. Sci. 1, 117–236 (2005)
6. Papadimitriou, S., Brockwell, A., Faloutsos, C.: Adaptive, hands-off stream mining. In: Int. Conf. on Very Large Data Bases(VLDB), pp. 560–571 (2003)
7. Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: ACM Symposium Principles of Database Systems(PODS), pp. 286–296 (2004)
8. Babcock, B., Olston, C.: Distributed topk monitoring. In: ACM International Conference on Management of Data (SIGMOD), pp. 28–39 (2003)
9. Frey, B., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)
10. Frey, B., Dueck, D.: Supporting online material of clustering by passing messages between data points. Science 315 (2007), http://www.sciencemag.org/cgi/content/full/1136800/DC1
11. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams: Theory and practice. IEEE Transactions on Knowledge and Data Engineering (TKDE) 15, 515–528 (2003)
12. Page, E.: Continuous inspection schemes 41, 100–115 (1954)
13. Hinkley, D.: Inference about the change-point from cumulative sum tests. Biometrika 58, 509–523 (1971)
14. Leone, M., Sumedha, W.M.: Clustering by soft-constraint affinity propagation: Applications to gene-expression data. Bioinformatics 23, 2708 (2007)
15. Ester, M.: A density-based algorithm for discovering clusters in large spatial databases with noisethe uniqueness of a good optimum for k-means. In: International Conference on Knowledge Discovery and Data Mining(KDD) (1996)
16. Keogh, E., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR time series classification/clustering homepage (2006), http://www.cs.ucr.edu/~eamonn/time_series_data/
17. KDD99: Kdd cup 1999 data (computer network intrusion detection) (1999), http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
18. Lee, W., Stolfo, S., Mok, K.: A data mining framework for building intrusion detection models. In: IEEE Symposium on Security and Privacy, pp. 120–132 (1999)
19. Dang, X.H., Ng, W.K., Ong, K.L.: An error bound guarantee algorithm for online mining frequent sets over data streams. Journal of Knowledge and Information Systems (2007)
20. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining highspeed data streams. In: ACM International Conference on Management of Data (SIGMOD), pp. 523–528 (2003)
21. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding hierarchical heavy hitters in streaming data. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(4) (2008)
22. Agarwal, D.K.: An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In: International Conference on Data Mining (ICDM) (2005)