# Statistical σ-Partition Clustering over Data Streams

Nam Hun Park and Won Suk Lee

Department of Computer Science, Yonsei University
134 Shinchon-dong Seodaemun-gu Seoul, 120-749, Korea
{zyonix,leewo}@amadeus.yonsei.ac.kr

**Abstract.** This paper proposes a grid-based clustering method that dynamically partitions the range of a grid-cell based on its distribution statistics of data elements in a data stream. Initially the multi-dimensional space of a data domain is partitioned into a set of mutually exclusive equal-size *initial cells*. As a new data element is generated continuously, each cell monitors the distribution statistics of data elements within its range. When the support of data elements in a cell becomes high enough, the cell is dynamically divided into two mutually exclusive smaller cells called *intermediate cells* by assuming the distribution of data elements is a normal distribution. Eventually, the dense sub-range of an initial cell is recursively partitioned until it becomes the smallest cell called *a unit cell*. In order to minimize the number of cells, a sparse intermediate or unit cell can be pruned if its support becomes much less than a minimum support. The performance of the proposed method is comparatively analyzed through a series of experiments.

## 1 Introduction

Recently, several data mining methods[1,2,3] for a data stream are actively proposed. A data stream is a massive unbounded sequence of data elements continuously generated at a rapid rate. Due to this reason, it is impossible to maintain all elements of a data stream. Consequently, data stream processing should satisfy the following requirements[4]. First, each data element should be examined at most once to analyze a data stream. Second, memory usage for data stream analysis should be restricted finitely although new data elements are continuously generated in a data stream. Third, newly generated data elements should be processed as fast as possible to produce the up-to-date analysis result of a data stream, so that it can be instantly utilized upon request. To satisfy these requirements, data stream processing sacrifices the correctness of its analysis result by allowing some errors.

This paper proposes a grid-based clustering method that dynamically partitions the range of a grid-cell based on its distribution statistics of data elements in a data stream. Initially the multi-dimensional space of a data domain is partitioned into a set of mutually exclusive equal-size initial cells. As a new data element is generated continuously, each cell monitors the distribution statistics of data elements within its range. When the support of a cell becomes high enough, the cell is dynamically divided into two mutually exclusive smaller cells, called *intermediate cells*, based on its distribution statistics. Similarly, a dense intermediate cell itself can be partitioned but it is replaced by its two-divided cells. Eventually, the dense sub-range of an initial cell is recursively partitioned until it becomes the smallest cell called a *unit cell*. A cluster of a data stream is a group of adjacent dense unit cells. As the size of a unit cell is set to be smaller, the resulting set of clusters is more accurately identified. In order to minimize the number of cells, a sparse intermediate or unit cell is pruned if its support becomes much less than a minimum support.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 presents the proposed statistical σ-partition clustering algorithm in detail. In Section 4, several experimental results are comparatively analyzed to illustrate the various characteristics of the proposed method. Finally, Section 5 presents conclusions.

## 2 Related Works

Clustering is a process of finding groups of similar data elements which are defined by a given similarity measure. Clustering techniques are categorized into several methods: partitioning, hierarchical, density-based and grid-based. The partitioning method such as k-means[5] and k-medoid[6] divides the data space of a data set into *k* mutually disjoint regions called clusters. The number of clusters should be predefined in advance. The k-medoid algorithm selects *k* data elements as the centers of *k* clusters initially, and repeatedly replaces one of the selected centers until it finds the best set of *k* centers. In this method, noise data elements can substantially influence the generation of a cluster, so that it may be difficult to produce a correct result in some cases. The hierarchical method such as BIRCH[7] and CURE[8] decomposes a data set into a tree-like structure. In BIRCH, a CF(Clustering Feature) tree which is used to summarize cluster representations is generated dynamically. After the CF tree is built, any clustering algorithm such as a typical partitioning algorithm is then used. In CURE, instead of using a single centroid to represent a cluster, a fixed number of well-scattered data objects is selected to represent a cluster. The selected representative data objects are shrunk towards the centroid of their cluster by a specified shrinking factor in the process of clustering. Among the clusters, two adjacent clusters whose representative data objects are the closest can be merged into one cluster until a predefined number of clusters is left. A typical density-based clustering algorithm[9] which regards a cluster as a region in a data space with a high density of data elements. Its strong points are that it can discover an arbitrarily shaped cluster, and control noise data easily. In the grid-based clustering method, the data space of a problem is divided statistically into a set of equal-size cells. A cluster is generated by merging adjacent cells that have more than a predefined number of data elements. Its time complexity is very efficient but the accuracy of a cluster is affected by the size of a cell. STING[10] uses a grid-based multi-resolution data structure in which a data space is divided into rectangular cells. There are several levels of such rectangular cells corresponding to the different levels of resolution.

Most conventional clustering algorithms assume a data set is fixed and focuses on how to minimize processing time or memory usage algorithmically. When a data set is enlarged incrementally, it is more efficient to use incremental clustering algorithms[7,11] which mainly focus on how to utilize the previous clustering result of an original data set in clustering its enlarged data set efficiently. In other words, the set of old data elements is scanned only when a new possible cluster may be found by the set of newly added data elements. Therefore, all the old data elements should be maintained physically.

In [13], a k-median algorithm is proposed to find the clusters of data elements generated in a data stream. It regards a data stream as a sequence of stream chunks. A stream chunk is a set of consecutive data elements generated in a data stream. Whenever a new stream chunk containing a set of newly generated data elements is formed, the LSEARCH routine which is an O(1)-approximate k-medoid algorithm is performed to select $k$ data elements from the data elements of the stream chunk as the local centers of the chunk. The algorithm confines its memory space to holding a fixed number of local centers for previous stream chunks. Therefore, if retaining $ik$ centers is impossible at the $i^{th}$ stream chunk, the LSEARCH routine is performed again to cluster the weighted $ik$ points to retain $k$ centers.

## 3 σ-Partition Clustering

Given a data stream $\boldsymbol{D}$ of d-dimensional data space $\mathbf{N}=\mathrm{N}_1\times \mathrm{N}_2\times \ldots \times \mathrm{N}_d$, a data element generated at the $j^{th}$ turn is denoted by $\boldsymbol{e}^j=<e_1^j,e_2^j,\ldots,e_d^j>$, $e_i^j \in \mathrm{N}_i$, $1\leq i\leq d$. When a new data element $\boldsymbol{e}^t$ is generated at the $t^{th}$ turn in a data stream $\boldsymbol{D}$, the current data stream $\boldsymbol{D}^t$ is composed of all the data elements that have ever been generated so far i.e. $\boldsymbol{D}^t=\{\mathbf{e}^1,\mathbf{e}^2,\ldots,\mathbf{e}^t\}$. The total number of data elements generated in the current data stream $\boldsymbol{D}^t$ is denoted by $|\boldsymbol{D}^t|$.

Finding a cluster of similar data elements in the current data stream $\boldsymbol{D}^t$ is identifying a region whose current density of data elements is dense enough. A unit cell whose length in each dimension is less than λ is used to define the similarity between data elements. The current support of a cell is the ratio of the number of those data elements in $\boldsymbol{D}^t$ that are inside the cell over the total number of data elements in $\boldsymbol{D}^t$. Therefore, a cluster at $\boldsymbol{D}^t$ is a group of adjacent dense unit cells whose current supports are greater than or equal to a predefined minimum support $S_{min}$.

The range of each dimension $\mathrm{N}_i$ is initially partitioned by $p$ number of mutually exclusive equal-size intervals $I_i^j=[s_i^j,f_i^j)$ $1\leq j\leq p$ where $s_i^j$ and $f_i^j$ denote the start and end values in the $j^{th}$ interval of the $i^{th}$ dimension. Consequently, $p^d$ number of initial cells are formed in $\boldsymbol{N}$ and each initial cell $g$ is defined by a set of d intervals $\{I_1,I_2,\ldots,I_d\}$ $I_i\subseteq \mathrm{N}_i$ $1\leq i\leq d$. The range R($g$) of an initial cell $g$ is a rectangular space $\boldsymbol{rs}=I_1\times\ldots\times I_d$. However, the initial rectangular space of an initial cell becomes a set of rectangular spaces $\boldsymbol{RS}=\{\boldsymbol{rs}_1,\boldsymbol{rs}_2,\ldots,\boldsymbol{rs}_q\}$ as a series of cell partitioning and pruning operations are performed subsequently. When these rectangular spaces are projected to the $i^{th}$ dimension, the intervals of the $i^{th}$ dimension of a cell g can be found and they are denoted by

$IS_i(g)=\{I_i^1, I_i^2, \ldots, I_i^q\}$. The sum of these intervals is defined as the interval size of the $i^{th}$ dimension of the cell $g$. The range of the cell $g$ is the united spaces of all the rectangular spaces $\boldsymbol{rs}_1,\ldots,\boldsymbol{rs}_q$, $R(g)=\bigcup_{i=1}^{q} rs_i$ . Each cell keeps the current distribution statistics of those data elements in the current data stream $\boldsymbol{D}^t$ that are within its range as defined in Definition 1.

**[Definition 1]** Distribution Statistics of a grid-cell $g(\boldsymbol{RS},c,\mu,\sigma)$
For the current data stream $\boldsymbol{D}^t$, a term $g(\boldsymbol{RS}, c^t, \mu^t, \sigma^t)$ is used to denote the distribution statistics of a cell $g$ which is defined by a set of its rectangular spaces $\boldsymbol{RS}$. Let $\boldsymbol{D}_g^t$ denote those elements in $\boldsymbol{D}^t$ that are in the range of the cell $g$, i.e.,$\boldsymbol{D}_g^t=\{\ \boldsymbol{e} \mid \boldsymbol{e}\in\boldsymbol{D}^t$ and $\boldsymbol{e} \in R(g)\ \}$. The distribution statistics of the cell $g$ are defined as follows:

i) $c^t$ : the number of data elements in $\boldsymbol{D}_g^t$

ii) $\mu^t=<\mu_1^t,\ldots,\mu_d^t>$ : $\mu_i^t$ denotes the average of the $i^{th}$ dimensional values of the data elements in $\boldsymbol{D}_g^t$.

$$\mu_i^t= \sum_{j=1}^{c^t} e_i^j \,/\, c^t \ , \ 1\le i\le d$$

iii) $\sigma^t=<\sigma_1^t,\ldots,\sigma_d^t>$ : $\sigma_i^t$ denotes the standard deviation of the $i^{th}$ dimensional values of the data elements in $\boldsymbol{D}_g^t$.

$$\sigma_i^t=\sqrt{\sum_{j=i}^{c^t}(e_i^j-\mu_i^t)^2/c^t} \ , \ 1\le i\le d$$

When a new data element $\boldsymbol{e}^t$ is generated in the current data stream $\boldsymbol{D}^t$, its corresponding initial cell among the $p^d$ initial cells is identified based on the initial partitions of the data space **N**. If the data element is in the range of the initial cell $g$ and the distribution statistics of the cell $g$ was updated most recently at the insert of the $v^{th}$ data element ($v\le t$), its statistics remain the same as $g(\boldsymbol{RS}, c^v,\mu^v,\sigma^v)$ and they are updated to $g(\boldsymbol{RS}, c^t,\mu^t,\sigma^t)$ as follow: for $\forall i$, $1\le i\le d$

$$c^t=c^v+1,\ \mu_i^t=\frac{\mu_i^v\times c^v+e_i^t}{c^t},\quad \sigma_i^t=\sqrt{\frac{c^v}{c^t}\times(\sigma_i^v)^2+\frac{(\mu_i^v)^2+(e_i^t)^2}{c^t}-(\mu_i^t)^2}$$

For the current data stream $\boldsymbol{D}^t$, the current support of an initial cell $g(\boldsymbol{RS}, c^t, \mu^t, \sigma^t)$ is defined by the ratio of its count over the total number of data elements generated so far ,i.e. $c^t/|\boldsymbol{D}^t|$. When the current support of the cell becomes the same as a *predefined split support* $S_{splt}(S_{splt}<S_{min})$, two intermediate cells $g_1$ and $g_2$ are created as the children of the initial cell. To split the range of the cell $g$, a dividing dimension is selected based on the distribution deviation of data elements in the cell $g$. Among the dimensions whose interval sizes for the cell $g$ are larger than $\lambda$, the one with the smallest standard deviation, say $\sigma_k^t$, is chosen as a dividing dimension. Based on the standard deviation $\sigma_k^t$ in the dividing dimension, the set of intervals in the dividing dimension k is partitioned into two sets of intervals. One contains those intervals that are within

the interval $[\mu_k-\sigma_k, \mu_k+\sigma_k)$ in which the 68 percentage of data elements in the cell *g* is assumed to be distributed according to a normal distribution. The other includes the remaining intervals. The rectangular spaces of the cell *g* are divided into two mutually exclusive sets by a statistical σ-partition method with respect to the two sets of intervals in the dividing dimension. These two sets of the rectangular spaces are assigned to the ranges of the two divided cells $g_1$ and $g_2$ respectively. If a rectangular space of the cell *g* includes $\mu_k^t$, the corresponding interval of the dividing dimension is actually divided. Figure 1 illustrates how to divide the rectangular spaces of a cell *g* $(\boldsymbol{RS},c,\mu,\sigma)$ in a two-dimensional data space.

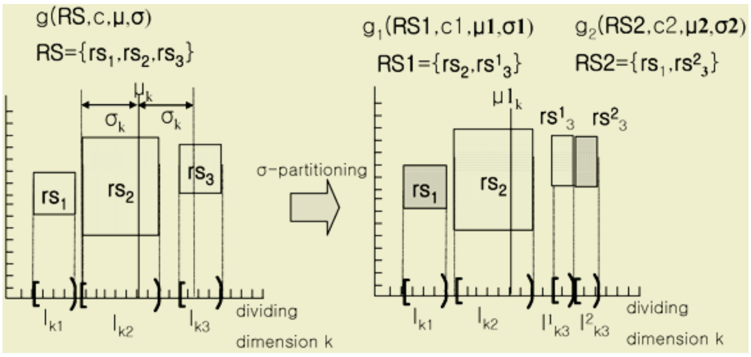

**Fig. 1.** σ-partition on a cell *g*

When a cell $g(\boldsymbol{RS}, c^t, \mu^t, \sigma^t)$ is partitioned by the above σ-partition method into cells $g_1(\boldsymbol{RS_1}, c1^t, \boldsymbol{\mu 1}^t, \boldsymbol{\sigma 1}^t)$ and $g_2(\boldsymbol{RS_2}, c2^t, \boldsymbol{\mu 2}^t, \boldsymbol{\sigma 2}^t)$, the distribution statistics of $g_1$ and $g_2$ are initialized as follows. Let $\varphi(x)=\frac{1}{\sqrt{2\pi}\sigma^t}e^{-\frac{(x-\mu^t)^2}{2(\sigma^t)^2}}$ be the normal distribution function of the data elements in the dividing dimension k for the cell *g*.

$$c1^t=c^t\times\int_{\mu_k^t-\sigma_k^t}^{\mu_k^t+\sigma_k^t}\varphi(x)dx\,,\ c2^t=c^t-c1^t \tag{1}$$

$$\boldsymbol{\mu 1}^t=\boldsymbol{\mu 2}^t=\mu^t \text{ except } \mu 1_k^t \text{ and } \mu 2_k^t\,,\ \ \mu 1_k^t=\int_{s_k(g_1)}^{f_k(g_1)}x\varphi(x)dx$$

$$\text{if } s_k(g_2)<\mu 1_k^t<f_k(g_2),\ \mu 2_k^t=\int_{s_k(g_2)}^{f_k(g_2)}x\varphi(x)dx-\mu 1_k^t$$

$$\text{else}\quad \mu 2_k^t=\int_{s_k(g_2)}^{f_k(g_2)}x\varphi(x)dx$$

$\boldsymbol{\sigma 1}^t = \boldsymbol{\sigma 2}^t = \sigma^t$ except $\sigma 1_k^{\,t}$ and $\sigma 2_k^{\,t}$, $\sigma 1_k^{\,t} = \sqrt{\int_{s_k(g_1)}^{f_k(g_1)} x^2 \varphi(x)dx - \left(\mu 1_k^{\,t}\right)^2}$ and

if $s_k(g_2) < \mu 1_k^{\,t} < f_k(g_2)$, $\sigma 2_k^{\,t} = \sqrt{\int_{s_k(g_2)}^{f_k(g_2)} x^2 \varphi(x)dx - \int_{s_k(g_1)}^{f_k(g_1)} x^2 \varphi(x)dx - \left(\mu 2_k^{\,t}\right)^2}$

$$\text{else } \sigma 2_k^{\,t} = \sqrt{\int_{s_k(g_2)}^{f_k(g_2)} x^2 \varphi(x)dx - \left(\mu 2_k^{\,t}\right)^2}$$

where $s_k(g_i)$ and $f_k(g_i)$ denote the smallest start and largest end value of the intervals of the $k^{th}$ dividing dimension for the divided cell $g_i$, $i$=1,2. At the same times, the distribution statistics of the original cell $g(\boldsymbol{RS}, c^t, \boldsymbol{\mu}^t, \boldsymbol{\sigma}^t)$ are reset as $c^t$=0 and $\mu_i^t = \sigma_i^t = 0$ for $\forall i$, $1 \leq i \leq$ d since they are carried to those of $g_1$ and $g_2$.

When a newly generated data element is not in the range of its corresponding initial cell, the children of the initial cell are searched to find the one whose range includes the element. After the target intermediate cell $g$ is found, its distribution statistics are updated by the same way as in an initial cell. When the updated support of the intermediate cell itself becomes the same as $S_{splt}$ and the range of the cell is larger than that of a unit cell, the intermediate cell $g$ is divided into two smaller intermediate cells by the same way of dividing an initial cell. As in an initial cell, among the dimensions whose interval sizes are larger than λ, the one with the smallest standard deviation is chosen as a dividing dimension. However, unlike an initial cell, the original intermediate cell is replaced by the two divided cells. Consequently, the parent initial cell of the original cell becomes the parent of each divided cell.

On the other hand, when the current support of an intermediate cell $g$ becomes less than a *predefined pruning support* $S_{prn}$ ,i.e., $c^t/|\boldsymbol{D}|^t < S_{prn}$, the probability of finding a cluster in the range of the cell in the near future is very low. Consequently, the cell is removed and its distribution statistics $g(\boldsymbol{RS}, c^t, \boldsymbol{\mu}^t, \boldsymbol{\sigma}^t)$ are returned back to its parent initial cell $g_p$. Suppose the distribution statistics of the parent cell $g_p$ were updated lastly at the $v^{th}$ element($v \leq t$) and they are denoted by $g_p(\boldsymbol{RSp}, cp^v, \boldsymbol{\mu p}^v, \boldsymbol{\sigma p}^v)$ where $\boldsymbol{\mu p}^v = <\mu p_1^{\,v}, \mu p_2^{\,v}, \ldots, \mu p_d^{\,v}>$ and $\boldsymbol{\sigma p}^v = <\sigma p_1^{\,v}, \sigma p_2^{\,v}, \ldots, \sigma p_d^{\,v}>$. Its new statistics $g_p(\boldsymbol{RSp}, cp^t, \boldsymbol{\mu p}^t, \boldsymbol{\sigma p}^t)$ at $\boldsymbol{D}^t$ is updated as follows:
For all dimensions $i(1 \leq i \leq \text{d})$, $cp^t = cp^v + c^t$ and

$$\mu p_i^{\,t} = \frac{\mu p_i^{\,v} \times c^v + \mu_i^{\,t} \times c^t}{cp^t} \text{ and}$$

$$\sigma p_i^{\,t} = \sqrt{\frac{cp^v \times (\sigma p_i^v)^2 + c^t \times (\sigma_i^t)^2}{cp^t} + \frac{(\mu p_i^v)^2 + (\mu_i^t)^2}{cp^t} - (\mu p_i^t)^2}$$

When a cell is divided, the 68% of its count is assigned to one divided cell and the rest, i.e., 32% is assigned to the other in Equation (1). Therefore, the value of a pruning support $S_{prn}$ should be less than the 32% of $S_{splt}$ in order to avoid pruning a newly divided cell too soon.

As mentioned, a sparse intermediate or unit cell can be pruned when a data element in the range of the cell is generated. However, a considerable number of such sparse cells may not be pruned since the possibility of encountering a data element in the range of a sparse cell is very low. All sparse intermediate or unit cells can be forced to be pruned together by examining their current supports. This mechanism is called as a *force-pruning operation*. Since the distribution statistics of all intermediate or unit cells should be examined, the processing time of a force-pruning operation takes relatively long. Due to this reason, it can be performed periodically or when the current number of cells reaches a predefined threshold value.

```
divide $N_1,\ldots,N_d$ into $p$ intervals and create $p^d$ initial cells;
 /*  $S(g)$ : the support of cell $g$ , $S(g)=c^t / |D^t|$  */

for a data stream $D^t=\{ e^1, e^2, \ldots, e^t \}$ do /* $t$ is enlarging */
   read current data element $e^t$;
   search the cell $g$ which includes $e^t$;
   update $\mu^t$, $\sigma^t$, $c^t$ of the cell $g$;
   if $g$ is a unit cell or an intermediate cell{
        if $S(g) >= S_{splt}$ {
              if $|f_i(g) - s_i(g)| > \lambda$ in any $i$ dimension {
                   /* dividing an intermediate cell $g$*/
                   find the largest $\sigma_k^t$ among $\sigma^t$ where $|f_k(g) - s_k(g)| > \lambda$;
                   generate $g_1$ and $g_2$;
                   set the statistics of $g_1$ and $g_2$; eliminate cell $g$;
              }
        }
         else if $S(g) <= S_{prn}$ {
              /* pruning a cell $g$ */
              find the parent initial cell $g_p$ where $g$ is included to $g_p$;
              update $g_p$ with statistics of $g$;
              eliminate cell $g$;
         }
   }
   else if $g$ is an initial cell {
         if $S(g) >= S_{splt}$ {
              /* dividing an initial cell $g$ */
              find the largest $\sigma_k^t$ among $\sigma^t$ where $|f_k(g) - s_k(g)| > \lambda$;
              generate $g_1$ and $g_2$;
              set the statistics of $g_1$ and $g_2$;
              set $c^t$=0 and $\mu_i^t=\sigma_i^t$=0 for $\forall i$ dimension;
         }
   }
end
```

**Fig. 2.** The statistic σ-partition clustering

Figure 2 shows the detailed steps of the proposed algorithm. When a cell is split, the counts of two divided cells are initialized by assuming the actual distribution of data elements in the dividing dimension of the cell is a normal distribution. However, if the actual distribution of data elements in the original cell is not the normal distribution, there is a certain estimation error. In other words, the count of each divided cell may be incorrectly initialized. This estimation error of a certain range in a data space is accumulated until the range is represented by a unit cell through a series of cell partitioning. Once the range becomes a unit cell, there is no additional estimation error since the number of data elements in its range is actually counted. Therefore, this accumulated error count of a unit cell is constant. However the support error of the accumulated error count in a dense unit cell is continuously decreased due to Property 1. For a new unit cell $g(\boldsymbol{RS}, c^t, \mu^t, \sigma^t)$ created in the current data stream $\boldsymbol{D}^t$, let $|\boldsymbol{D}_g^t|$ denote the actual count of data elements in its range R($g$) up to $\boldsymbol{D}^t$. The estimation error E($g$) in the count $c^t$ of the cell $g$ is constant and is defined by the difference between $|\boldsymbol{D}_g^t|$ and its estimated count $c^t$, i.e., E($g$)= $|\,|\boldsymbol{D}_g^t| - c^t|$.

**Property 1. (Support error decreasing property)** When a unit cell $g$ is newly created in the current data stream $\boldsymbol{D}^t$, its support error is E($g$)/$|\boldsymbol{D}|^t$ and the estimation error E($g$) is constant. After $m$ new additional data elements are processed subsequently, the total number of data elements is increased to $|\boldsymbol{D}^{t+m}|$ in $\boldsymbol{D}^{t+m}$ and $|\boldsymbol{D}^{t+m}| > |\boldsymbol{D}^t|$ is satisfied. Consequently, the support error of the cell $g$ in $|\boldsymbol{D}^{t+m}|$ becomes $\frac{\mathrm{E}(g)}{|\boldsymbol{D}^{t+m}|}$, and $\frac{\mathrm{E}(g)}{|\boldsymbol{D}^{t+m}|} < \frac{\mathrm{E}(g)}{|\boldsymbol{D}^{t}|}$ is satisfied. As $m$ is increased infinitely, $\frac{\mathrm{E}(g)}{|\boldsymbol{D}^{t+m}|}$ converges to 0, i.e. $\lim_{m\to\infty} \frac{\mathrm{E}(g)}{|\boldsymbol{D}^{t+m}|} \approx 0$. Therefore, it can be ignorable.

A unit cell in the current data stream $\boldsymbol{D}^t$ is dense if its current support is greater than or equal to a predefined minimum support $S_{min}$. A cluster in the current data stream is a set of adjacent dense unit cells. As the size of a unit cell $\lambda$ is defined to be smaller, the range of a cluster is more precisely identified. On the other hand, a possible dense cell is split earlier as the value of a split support $S_{splt}$ is lower. Due to this reason, a dense unit cell is found earlier, which enables a unit cell to monitor its actual count more accurately. Furthermore, as the gap between $S_{prn}$ and $S_{splt}$ is increased, more number of intermediate cells are maintained while the support of identified clusters are more precisely found.

## 4 Experimental Results

In order to analyze the performance of the proposed method, a data set containing one million 4-dimensional data elements is generated by the data generator used in ENCLUS [14]. Most of data elements are concentrated on randomly chosen 10 distinct data regions whose sizes in each dimension are also randomly varied from 10 to 20 respectively. The result of the proposed method is compared with that of the grid-based clustering algorithm STING. The values of a pruning support $S_{prn}$ and a split support $S_{splt}$ are assigned relatively to a predefined minimum support $S_{min}$. The entire data space of the data set is divided into 4 initial cells. In all experiments, data elements are looked up one by one in sequence to simulate the environment of a data stream.

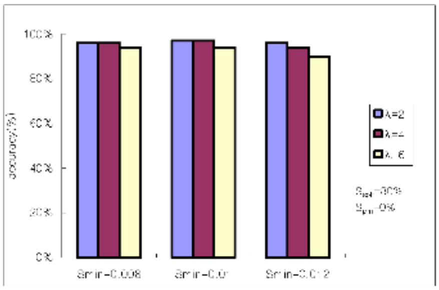

**Fig. 4.** Accuracy variations to $S_{min}$ and λ

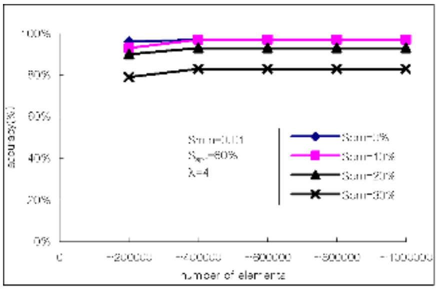

**Fig. 5.** Accuracy variations to $S_{prn}$

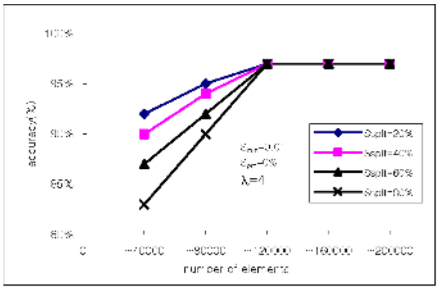

**Fig. 6.** Accuracy variations to $S_{splt}$

Figure 4 shows the accuracy of the proposed method by varying the values of λ and $S_{min}$. The accuracy of the proposed method is measured relatively to that of STING. In other words, it is the ratio of the number of correctly clustered elements by the proposed method over the total number of data elements clustered by STING.

Figure 5 shows the accuracy of the proposed method by varying the value of $S_{prn}$. The sequence of generated data elements is divided into 5 intervals each of which consists of 200 thousand elements. The average accuracy in each interval is shown. As noticed in this figure, the accuracy of the first interval is relatively lower than those of the other intervals. This is because the support of an intermediate cell is too sensitively varied in the first interval. As a result, a lot of cell partitioning operations are performed in the first interval to produce a set of meaningful unit cells. However, it becomes stabilized as the total number of data elements is increased. As the value of $S_{prn}$ is increased, the accuracy becomes lower since a considerable number of possible dense intermediate cells are pruned before they become unit cells. Figure 6 shows the effect of $S_{splt}$ on the accuracy in the first interval. As the value of $S_{splt}$ is set to be lower, unit cells are generated more quickly, so that the accuracy is improved in the early stage of clustering. However, regardless of the values of $S_{splt}$, the stabilized accuracy is the same.

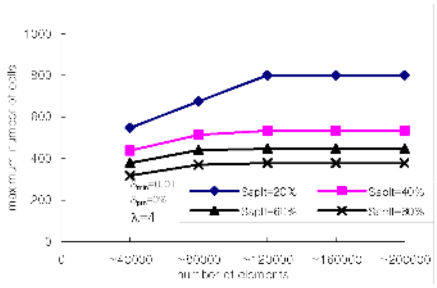

**Fig. 7.** Memory usage variations to $S_{splt}$

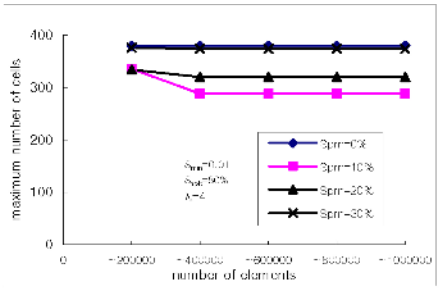

**Fig. 8.** Memory usage variations to $S_{prn}$

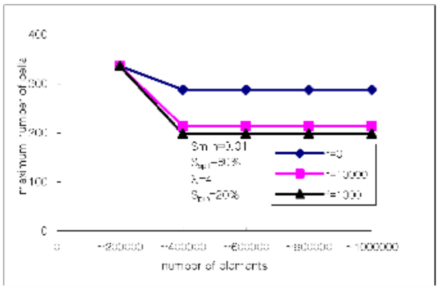

**Fig. 9.** Memory usage with force-pruning

Figure 7 shows the maximum number of cells in the first interval when no cell is pruned. The maximum number of cells is stabilized after the first interval. As the value of $S_{splt}$ is set to be low, the maximum number of cells is increased since cell partitioning operations are performed frequently to generate meaningful unit cells. As the number of dense unit cells is increased, the maximum number of cells is stabilized. In Figure 8, the variation of the maximum number of cells is shown when sparse cells are pruned. After most of dense unit cells are generated, the maximum number of cells can be decreased by setting the value of $S_{prn}$ adequately. When $S_{prn}$ is set to the 30% of $S_{splt}$, the memory usage is not decreased. The reason is that most of divided intermediate cells are pruned too quickly and their initial cells are repeatedly partitioned again. On the contrary, when the value of $S_{prn}$ is set to 10%, the memory usage is minimized since dense intermediate cells are successfully divided into its dense unit cells while sparse ones are pruned properly.

A force-pruning operation is usually performed periodically or when it is needed. Figure 9 shows the memory usage of the proposed method by varying the period of a force-pruning operation. In this experiment, two force-pruning periods f=1,000 and f=10,000 are compared. A force-pruning period f=1,000 means that a force-pruning operation is performed whenever 1,000 new data elements are processed. The memory usage of each interval is represented by the maximum number of cells. The number of cells is decreased as the period is shortened. As noticed by this experiment, a force pruning operation does not have to be performed frequently. Instead, it only needs to be performed when lots of intermediate cells are partitioned.

## 5 Conclusion

In this paper, a grid-based statistical σ-partition clustering method for a data stream is proposed. The multi-dimensional data space is dynamically divided into a set of cells with different sizes. By maintaining only the distribution statistics of data elements in each cell, its current support is precisely monitored. A dense sub-range of a data space is partitioned repeatedly until it becomes a set of dense unit cells. Two thresholds $S_{splt}$ and $S_{prn}$ are proposed to control the performance of the proposed method in a data stream. A split support $S_{splt}$ is used to determine how fast dense unit cells are identified. A pruning support $S_{prn}$ is used to remove meaningless sparse intermediate or unit cells. Therefore, it can be used to minimize the usage of main memory. However, if it is too high, a less accurate clustering result can be obtained. By controlling these two thresholds properly, the performance of the proposed algorithm can be flexibly controlled.

## References

1. M. Datar, A. Gionis, P. Indyk and R. Motwani. Maintaining stream statistics over sliding windows. In *Proc. Of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms, Jan.* 2002
2. M. Charikar, K. Chen and M. Farach-Colton. Finding Frequent Items In Data Streams. In *Proc. Of the 29th Int'l Colloq. on Automata, Language and Programming,* 2002
3. G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. Of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, Aug. 2002.
4. M. Garofalakis, J. Gehrke and R. Rastogi. Querying and mining data streams: you only get one look. In *the tutorial notes of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, Aug. 2002.
5. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1972.
6. L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis.* Wiley, New York, 1990.
7. T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. SIGMOD*, pages 103-114, 1996
8. S. Guha, R.Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. SIGMOD*, pages 73-84, 1998
9. M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases, 1996.
10. W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining, 1997.
11. M. Ester, H. Kriegel, J. Sander, M. Wimmer, and X. Xu. Incremental clustering for mining in a data warehousing environment, In Proc. VLDB 24th, New York, 1998

12. G. S. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of the 28th Int'l Conference on Very Large Databases,* Hong Kong, China, Aug. 2002.
13. Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. STREAM-data algorithms for high-quality clustering. In *Proc. of IEEE International Conference on Data Engineering*, March 2002.
14. Cheng, C., Fu, A., and Zhang, Y. *Entropy-based subspace clustering for mining numerical data*. KDD-99, 84-93, San Diego, August 1999.