실시간 데이터 스트림 분석을 위한 클러스터링 기법

박남훈, 이원석
연세대학교 컴퓨터과학과
e-mail : {zyonix, leewo}@database.yonsei.ac.kr

A Grid-based Clustering Method for a Data Stream

Nam Hun Park, Won Suk Lee
Dept. of Computer Engineering, Yonsei University

요 약

데이터 스트림이란, 빠른 속도로 지속적으로 생성되는 무한한 크기의 방대한 양의 데이터 집합으로 정의된다. 본 연구에서는 데이터 스트림 분석을 위한 데이터 스트림 격자 기반 클러스터링 기법을 제시한다. 주어진 초기 격자셀에 대해, 데이터 객체의 빈도가 높은 범위를 반복적으로 보다 작은 크기의 격자셀로 분할하여 최소 크기의 격자셀, 단위 격자셀을 생성한다. 격자셀에서는 데이터 객체들의 분포에 대한 통계값만을 저장하여, 기존의 클러스터링 기법에 비해 데이터 객체에 대한 탐색없이 효율적으로 클러스터를 찾을 수 있다.

1. 서론

최근, 웹 서핑 로그나 통신 로그, 실시간 동영상 데이터, 실시간 주식 거래 등과 같이 방대한 실시간 데이터를 처리하는 응용 환경이 증가함에 따라 이러한 데이터 스트림 분석에 대한 수요가 증가함에 따라 데이터 스트림 분석을 위한 데이터 마이닝 연구[1,2,3,4,5,6,7,8]들이 진행되어 왔다. 데이터 스트림이란, 빠른 속도로 지속적으로 생성되는 무한한 크기의 방대한 양의 데이터 집합으로 정의된다. 따라서, 기존의 대부분 분석방법처럼 데이터 스트림을 저장하고 반복 분석하는 것은 불가능하기 때문에, 데이터 스트림 처리 기법들은 다음과 같은 제약조건들을 충족시켜야 한다. 첫째, 각 데이터 객체들은 최대 한번의 검색으로 분석 될 수 있어야 한다. 둘째, 무한한 데이터 스트림에 비해 이를 분석하는 메모리 공간은 한정되어 있다. 셋째, 최근 생성된 데이터 객체는 최대한 빠르게 분석되어, 앞으로의 분석작업에 그 결과를 반영할 수 있어야 한다. 이러한 제약조건 내에서 데이터 스트림 분석 작업은 일정 한도내의 오차를 허용하기도 한다. 클러스터링은 데이터 마이닝 분석 방법 중 하나로서, 주어진 유사도 척도에 따라 다수의 데이터 객체들을 유사한 집단으로 구분하는 데이터 마이닝 기법이다. 기존의 대부분 클러스터링 기법[7,8,9,10]들은 한정된 데이터 집합을 대상으로 분석 결과의 정확도를 유지하면서 수행시간과 메모리 사용량을 최소화 하였다. 비록 클러스터링 기법들이 많은 응용분야에서 활용되고 있지만, 방대한 양의 데이터 집합을 대상으로 수행할 수 있는 클러스터링 기법은 많지 않다. 점진적으로 증가하는 데이터 집합을 대상으로 점진적 클러스터링 기법(Incremental Clustering Algorithm)[8,11]들이 제시되었다. 이들 연구들은 이전 분석 결과들을 적용하여 새로 생성된 데이터 집합에 영향을 받는 최근 클러스터를 효율적으로 찾는 방법을 제시하였다. 하지만, 이전 데이터 집합을 모두 저장되어, 일부 데이터들을 반복 분석해야하기 때문에, 데이터 스트림 환경에서는 적합하지 못하다. [12]의 연구에서는 데이터 스트림을 대상으로 분할 클러스터링(Partitioning Clustering)기반의 분석 방법을 제시하였다. 분할된 데이터 스트림의 각 부분 스트림들을 대상으로 O(1)-approximate K-medoid 를 수행하였으며, 기존 K-

이 논문은 0000년도 정부(과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업으로 수행된 연구임
(No.M10600000225-06J0000-22510)

medoid[7] 기법들이 데이터 집합을 반복해서 분석하는 것에 비해 각 부분 스트림들의 k 개의 중심점을 지속적으로 유지한다.

본 연구에서는 데이터 스트림 분석을 위한 격자 기반 클러스터링 기법을 제시한다. 정밀하게 분할된 각 격자셀에 위치한 데이터 객체들의 통계적 분포정보만을 지속적으로 유지하여 데이터 객체들을 저장하지 않으면서 한번의 검색으로 클러스터를 찾을 수 있다. 먼저, 데이터 스트림의 다차원 데이터 공간을 중첩되지않는 동일한 크기의 초기 격자셀로 분할한다. 데이터 객체가 생성됨에 따라 각 초기 격자셀은 해당 범위 내의 데이터 객체들의 통계적 분포정보를 갱신한다. 초기 격자셀의 지지도가 높은 경우, 데이터 객체들의 통계적 분포정보를 기반으로 분할차원을 선택하여 이를 기준으로 두 개의 격자셀, 중간 격자셀을 생성한다. 중간 격자셀 내의 데이터 객체들의 통계적 분포 정보는 기존 격자셀의 분포 정보로부터 추정할 수 있다. 이와 같이, 데이터 객체의 빈도가 높은 범위의 중간 격자셀들은 반복적으로 보다 작은 크기의 중간 격자셀로 분할하게 되며, 최종적으로 최소 크기의 격자셀, 단위 격자셀이 생성된다.

본 논문은 다음과 같이 구성된다. 2 장에서는 관련된 기존 연구들에 대해서 설명을 하고, 3 장에서는 본 논문에서 제안하는 격자 기반의 데이터 스트림 클러스터링 방법을 설명한다. 4 장에서는 실험을 통해 제안한 방법의 정확도 및 성능을 분석하고, 마지막으로 5 장에서 결론을 제시한다.

2. 격자기반 데이터 스트림 클러스터링

d 차원 데이터 공간 $\boldsymbol{N}=N_1\times N_2\times\ldots\times N_d$ 에서 주어진 데이터 스트림 $\boldsymbol{D}$ 내의 j 번째 생성된 데이터 객체는 $\boldsymbol{e}^j=<e_1^j,e_2^j,\ldots,e_d^j>$, $e_i^j\in N_i$, $1\le i\le$ d 로 나타낸다. t 번째 차례에 $\boldsymbol{e}^t$ 가 생성된 경우 최근 데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$ 는 t 번째까지 생성된 데이터 객체들의 집합 $\boldsymbol{D}^{\mathrm{t}}=\{\boldsymbol{e}^1,\boldsymbol{e}^2,\ldots,\boldsymbol{e}^t\}$으로 정의하며, 최근 데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$ 내의 데이터 객체의 수는 $|\boldsymbol{D}^{\mathrm{t}}|$으로 표기한다. 데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$ 의 클러스터링은 데이터 객체의 빈도가 높은 지역을 찾는 방법으로 가정할 수 있다. 따라서, 데이터 객체간의 유사도를 기준으로 삼는 기존의 클러스터링 기법과 유사하게 격자기반 클러스터링에서는 사용자 정의 거리 λ 를 기준으로 모든 차원축 상의 범위가 λ 이하인 격자셀을 단위 격자셀로 정의하고, $\boldsymbol{D}^t$ 내의 데이터 객체의 수에 대한 격자셀 내의 데이터 객체의 수의 비율을 격자셀의 지지도로 정의한다. 데이터 스트림 $\boldsymbol{D}^t$ 의 클러스터는 지지도가 주어진 최소지지도 S_{min} 이상인 인접한 단위 격자셀들의 집합으로 정의된다.

먼저, 각 차원축 N_i 를 주어진 p 개의 동일한 크기의 범위 $I_i^j=[s_i^j,f_i^j)$ $1\le j\le p$, 로 나눈다. s_i^j 와 f_i^j 는 i 차원상의 j 번째 범위의 시작좌표와 끝좌표를 나타낸다. 즉, 각 초기 격자셀은 d 개의 범위가 $\{I_1,I_2,\ldots,I_d\}$ $I_i\subseteq \mathrm{N_i}$ $1\le i\le$ d 로 표시되며, 초기 격자셀 g 의 범위 $R(g)$는 각 범위가 교차하는 육방형 공간 $\boldsymbol{rs}=I_1\times\ldots\times I_d$ 로 정의된다. 결과적으로 다차원 데이터 공간 $\boldsymbol{N}$ 은 p^d 개의 초기 격자셀로 구성된다. 초기 격자셀의 범위는 이 후 분할과정을 수행함에 따라 육방형 공간의 집합 $\boldsymbol{RS}=\{\boldsymbol{rs}_1,\boldsymbol{rs}_2,\ldots,\boldsymbol{rs}_q\}$으로 정의된다. 이 때, 격자셀 g 의 i 번째 차원축에 대한 범위는 $IS_i(g)=\{I_i^1,I_i^2,\ldots,I_i^q\}$로 나타낸다. 즉, i 번째 차원에 대한 격자셀 g 의 크기 $|IS_i(g)|$는 이들 범위의 합으로 정의하며, 다차원 공간에서의 격자셀 g 의 범위는 육방형 공간 $\boldsymbol{rs}_1,\ldots,\boldsymbol{rs}_q$ 의 합, $R(g)=\bigcup_{i=1}^{q} rs_i$ 로 나타낸다. 각 격자셀은 해당 공간 내의 데이터 객체들의 분포에 대한 통계정보를 저장하며, 정의 1 과 같이 정의할 수 있다.

정의 1. 격자셀 $\boldsymbol{g(RS,c,\mu,\sigma)}$
최근 데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$에 대해, $g(\boldsymbol{RS}, c^t, \boldsymbol{\mu}^t, \boldsymbol{\sigma}^t)$는 격자셀 g 의 공간 $\boldsymbol{RS}$ 내의 데이터 객체들의 분포에 대한 통계정보를 나타낸다. 격자셀 g 내의 데이터 객체들의 집합 $\boldsymbol{D}_{\mathrm{g}}^{\mathrm{t}}=\{\ \boldsymbol{e}|\ \boldsymbol{e}\in\boldsymbol{D}^{\mathrm{t}}$ and $\boldsymbol{e}\in \mathrm{R}(g)\ \}$에 대해, 격자셀 g 의 통계정보는 다음과 같이 정의된다.

i) $c^t:\boldsymbol{D}_{\mathrm{g}}^{\mathrm{t}}$ 내의 데이터 객체의 수

ii) $\boldsymbol{\mu}^t=<\mu_1^t,\ldots,\mu_d^t>$: $\boldsymbol{D}_{\mathrm{g}}^{\mathrm{t}}$ 내의 데이터 객체들의 i 번째 차원에서의 평균값 μ_i^t

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

iii) $\boldsymbol{\sigma}^t=<\sigma_1^t,\ldots,\sigma_d^t>$: $\boldsymbol{D}_{\mathrm{g}}^{\mathrm{t}}$ 내의 데이터 객체들의 i 번째 차원에서의 표준편차 σ_i^t

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

□

최근 데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$ 에 생성된 데이터 객체 $\boldsymbol{e}^t$ 에 대해서 p^d 초기 격자셀 중 이를 포함하는 초기 격자셀 g 를 검색한다. 이전 $v(v\le t)$번째 생성된 데이터 객체에 의해 갱신된 격자셀 $g(\boldsymbol{RS}, c^v,\boldsymbol{\mu}^v,\boldsymbol{\sigma}^v)$는 $\boldsymbol{e}^t$에 의해 다음과 같이 $g(\boldsymbol{RS}, c^t,\boldsymbol{\mu}^t,\boldsymbol{\sigma}^t)$로 갱신된다.

$$c^t=c^v+1, \qquad \text{(식 1)}$$

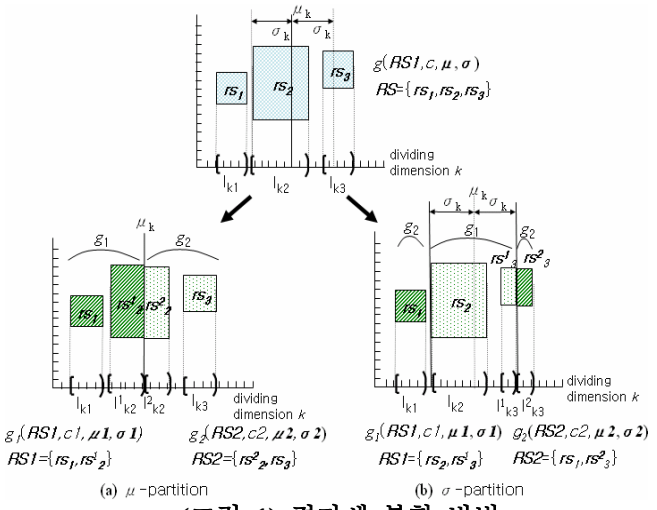
$$\mu_i^t=\frac{\mu_i^v\times c^v+e_i^t}{c^t}\ ,\ \sigma_i^t=\sqrt{\frac{c^v}{c^t}\times(\sigma_i^v)^2+\frac{(\mu_i^v)^2+(e_i^t)^2}{c^t}-(\mu_i^t)^2}$$

$$\text{for}\ \forall i,\ 1\le i\le \mathrm{d} \qquad \text{(식 2)}$$

데이터 스트림 $\boldsymbol{D}^{\mathrm{t}}$ 에 대해서, 격자셀 $g(\boldsymbol{RS}, c^t, \boldsymbol{\mu}^t, \boldsymbol{\sigma}^t)$의 지지도는 $\boldsymbol{D}^{\mathrm{t}}$ 내의 데이터 객체의 수에 대한 격자셀 내의 데이터 객체의 수의 비율, $c^t/|\boldsymbol{D}^t|$로 정의된다. 격자셀의 지지도가 주어진 분할지지도 $S_{splt}(S_{splt}<S_{min})$ 이상인 경우, 해당 공간 내에 클러스터가 존재하는 것으로 간주하고 보다 정확한 범위를 찾기 위해 해당 격자셀을 두 개의 중간 격자셀 g_1 과 g_2 를 해당 초기 격자셀의 하위로 생성되며, 이 때 생성되는 중간 격자셀들의 범위와 통계정보는 사용된 분할방법에 따라 결정된다. 분할방법은 3.2 장에서 자세히 기술한다.

초기 격자셀의 하위로 중간 격자셀이 존재하는 경우, 이들 중 새로 생성된 데이터 객체 e^{t}를 포함하는 중간 격자셀 g 를 검색하여 이의 통계정보를 갱신한다. 해당 중간 격자셀의 지지도가 S_{splt} 이상인 경우 마찬가지로 분할과정을 수행하게 되며, 초기 격자셀과 달리 분할된 중간 격자셀은 보다 작은 중간 격자셀들로 대체되어 해당 초기 격자셀의 하위에 위치한다.

본 연구에서는 격자셀의 분할방법으로는 μ-partition[13], σ-partition[14]과 hybrid-partition 방법을 제안한다. 먼저, 다차원 데이터 공간 N 에서의 각 차원들 중 격자셀 내의 데이터 객체들의 분포에 따라 클러스터를 효율적으로 찾을 수 있도록 분할차원(Dividing Dimension)을 결정하고, 선택된 분할차원 상의 범위를 분할한다. 분할방법에 따른 분할차원 선택방법은 각각 다르게 정의된다. μ-partition 은 격자셀 내의 데이터 객체들을 가능한 균등하도록 양분하는 분할방법이며, σ-partition 은 객체의 빈도가 높은 격자셀과 그 이외의 격자셀을 차등하도록 나누는 방법이다. 그리고, hybrid-partition 방법은 앞의 두 방법 중 데이터 객체들의 분포를 기반으로 보다 효율적인 방법을 동적으로 선택하는 방법이다.



(그림 1) 격자셀 분할 방법

본 클러스터링 방법에서는 적은 수의 분할과정으로 단위 격자셀을 빠르게 찾는 것이 본 성능을 결정하는 중요한 요소가 된다. 따라서, hybrid-partition 방법을 통해 두 분할방법 중 보다 단위 격자셀을 빨리 찾을 수 있는 방법을 선택하여 클러스터링에 필요한 분할과정을 줄일 수 있다. 이를 도시하면 그림 1 과 같다. 데이터 객체들이 균일하게 분포하는 i 차원의 표준편차 σ^e_i 에 대해서, 분할임계값 $\beta_k(g)$는 격자셀 g 의 표준편차와 σ^e_i 와의 차이로 다음과 같이 정의할 수 있다.

$$\beta_k(g) = |\sigma^e_k - \sigma^t_k|,$$

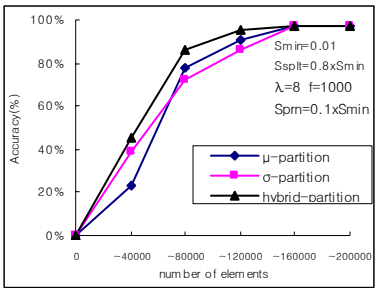
$$\text{where} \quad \sigma^e_k = \sqrt{\frac{1}{f_k(g) - s_k(g)} \int_{s_k(g)}^{f_k(g)} x^2 dx - (\mu^t_k)^2}$$

주어진 격자셀 g 에 대해서, μ-partition 과 σ-partition 에 의해 선택된 분할차원을 각각 *k1* 과 *k2* 라 간주한다. 즉, 데이터 객체들의 표준편차가 가장 큰 차원을 *k1*, 가장 작은 차원을 *k2* 라 한다. 이 때, 각 분할차원에 대한 분할임계값을 비교하여, $\beta_{k1}(g) > \beta_{k2}(g)$인 경우 데이터 객체들이 *k1* 차원에서 분산된 정도가 *k2* 차원에서 응집된 정도보다 큰 것으로 판정하고 μ-partition 을 선택해서 수행할 수 있다. 반대로, $\beta_{k1}(g) < \beta_{k2}(g)$인 경우, σ-partition 방법을 선택하여 효율적으로 분할과정을 수행할 수 있으며, $\beta_{k1}(g) = \beta_{k2}(g)$의 경우에는 분할임계값이 동일하기 때문에 둘 중 어느 방법으로도 분할을 수행할 수 있다. 데이터 객체의 분포에 따라 효율적인 분할방법을 선택하여 클러스터링 수행 동안의 분할과정의 수를 줄이고, 불필요한 격자셀의 생성을 방지할 수 있다.

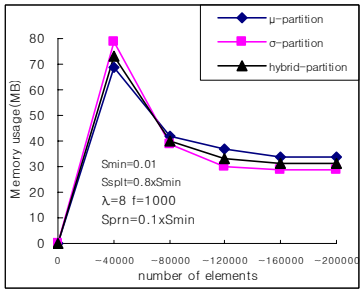
3. 실험 결과

본 데이터 스트림 클러스터링 방법의 성능을 분석하기 위해서, ENCLUS[15]의 데이터 생성기를 사용하여 40 차원의 백만개의 데이터 객체들로 구성된 데이터 스트림을 생성하였다. 데이터 객체들은 임의의 10 개의 영역에 밀집하여 생성되었으며, 밀집된 영역의 크기는 5~20 으로 생성되었다. 본 클러스터링 방법의 정확도는 기존의 대표적인 격자기반 클러스터링 기법인 STING 과 비교하였다. 즉, 전체 데이터 객체들의 수에 대해 본 클러스터링 수행결과 STING 과 동일한 클러스터로 판별된 데이터 객체들의 수의 비율로 정확도를 표시하였다. 주어진 다차원 데이터 공간은 4 개의 초기 격자셀로 나누었으며, 모든 실험에서 데이터 스트림 환경을 모의실험하기 위해 데이터 객체는 하나씩 순차적으로 처리되었다.

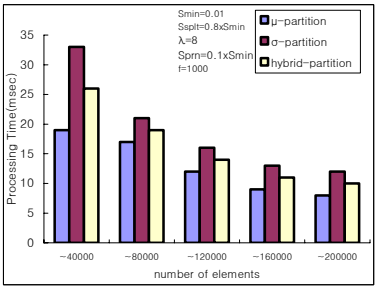
그림 2 는 각 분할방법에 따른 정확도의 변화를 도시하였다. 전체 데이터 객체의 수에 대한 STING 과 동일한 클러스터로 분류된 데이터 객체의 수의 비율을 도시하였으며, 데이터 스트림은 200,000 개 구간으로 구분하였으며, 격자셀의 분할이 잦은 첫 200,000 개 구간을 도시하였다. 강제전지과정은 40,000 개의 데이터 객체를 주기로 수행하였다. 클러스터링 시작단계에서는 각 격자셀의 지지도가 불안정하게 변화하므로 이후단계에 비해 정확도가 아주 낮지만, 분할과정 결과 단위 격자셀을 생성하면서 점차적으로 정확도가 높아지고, 성능이 안정화된다. 클러스터링 첫번째 구간에서는 hybrid-partition 방법이 가장 높은 정확도를 보인다. 격자셀 내의 데이터 객체들의 분포를 반영하여 분할방법을 선택하기 때문에 적은 수의 분할과정으로 단위 격자셀을 타방법보다 빠르게 생성하기 때문이다. 그림 3 은 첫번째 구간에서의 각 분할방법에 따른 메모리 사용량의 변화를 도시하였다. 최대 메모리 사용량을 비교하기 위해서 메모리 공간은 한정되지 않은 것으로 간주하였으며, 각 분할방법의 메모리 사용량은 40,000 개로 세분화된 각 구간에서의 최대 메모리 사용량을 도시하였다. 각 분할방법에 따라 측정된 최대 메모리 사용량은 미세한 차이를 보여준다. 그림 4 는 각 분할방법의 평균수행시간을 도시하였다. 최근 생성된 데이터 객체에 대해 해당 격자셀에서의 분할과정 또는 전지과정까지의 수행시간들의 각 구간에서의 평균수행시간으로 도시하였다. 첫번째 구간에서는 빈발한 분할과정으로 인해 평균수행시간이 크게 도시되었지만, 이후 성능이 안정화되면서 수행시간은 감소되어 동일하게 유지된다.



(그림 2) 클러스터링 정확도



(그림 3) 메모리 사용량



(그림 4) 수행시간

4. 결론

본 논문에서는 데이터 스트림 분석을 위한 격자기반 데이터 스트림 클러스터링 방법을 제시하였다. 다차원 데이터 공간을 그 분포에 따라 동적 범위의 격자셀들로 분할하며, 각 격자셀에서는 데이터 객체들의 분포에 대한 통계값만을 저장하여 기존의 클러스터링 기법에 비해 데이터 객체의 저장없이 효율적으로 수행할 수 있다. 데이터 객체의 밀도가 높은 범위는 단위 격자셀이 생성될 때까지 반복해서 분할하게 되며, 클러스터는 연속한 최소지지도 이상인 단위 격자셀들을 탐색하여 찾을 수 있다. 격자셀의 분할을 위해 μ-partition, σ-partition 과 hybrid-partition 분할방법을 제시하였다. μ-partition 와 σ-partition 분할방법은 고정된 비율로 격자셀을 분할하는 반면, hybrid-partition 방법은 동적으로 보다 효율적인 분할방법을 선택적으로 수행하여 데이터 객체의 분포를 분할과정에 반영할 수 있도록 하였다. 또한, 동적 메모리 관리 방법을 통해 적응적으로 λ 를 조절하여 제한된 메모리 공간을 최대한 사용하여 클러스터의 정확도를 주어진 자원에 따라 적응적으로 극대화하였다.

참고문헌

[1] 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.

[2] 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.

[3] J. H. Chang & W. S. Lee. Finding Frequent Itemsets over Online Data Streams. *Information and Software Technology*, 48(7), July 2006.

[4] J. H. Chang & W. S. Lee. Finding Recently Frequent Itemsets Adaptively over Online Transactional Data Streams. *Information Systems*, 31(8), December 2006

[5] Hua-Fu Li, Suh-Yin Lee, Man-Kwan Shan: Online Mining Changes of Items over Continuous ppend-only and Dynamic Data Streams. J. UCS 11(8), page 1411-1425, 2005

[6] Mohamed Medhat Gaber, Arkady B. Zaslavsky, Shonali Krishnaswamy: Mining data streams: a review. SIGMOD Record 34(2), page 18-26, 2005

[7] L. Kaufman and P.J. Rousseeuw. Finding Groups in Data. An Introduction to Cluster Analysis. Wiley, New York, 1990.

[8] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proc. SIGMOD*, pages 103-114, 1996

[9] S. Guha, R.Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. SIGMOD*, pages 73-84, 1998

[10] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases, 1996.

[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] 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.

[13] Nam Hun Park and Won Suk Lee. A statistical μ-partitioning method for clustering data streams. In *Proc. of Eighteenth International Symposium on Computer and Information Sciences*, November 2003.

[14] Nam Hun Park and Won Suk Lee. Statistical σ-partition Clustering over Data Streams. In *Proc. of 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, September 2003.

[15] Cheng, C., Fu, A., and Zhang, Y. Entropy-based subspace clustering for mining numerical data. KDD-99, 84-93, San Diego, August 1999