Journal of KIIT. Vol. 20, No. 1, pp. 49-55, Jan. 31, 2022. pISSN 1598-8619, eISSN 2093-7571 49
http://dx.doi.org/10.14801/jkiit.2022.20.1.49

슬라이딩 윈도우 페인을 이용한 데이터 스트림 클러스터링

박 남 훈*

Clustering Data Steams with Sliding Window Panes

Namhun Park*

요 약

실시간 데이터 스트림 환경에서 클러스터링 분석은 과거 정보의 영향으로 최근 데이터에 의한 분석 결과를 효율적으로 반영하지 못한다. 클러스터링 분석에서 중요하지 않은 과거 정보가 반영된 결과를 찾는 경우가 발생하는 것이다. 이러한 환경에서 과거의 정보를 구분하여 처리하기 위한 방법으로 슬라이딩 윈도우 기법이 있다. 본 논문에서는 슬라이딩 윈도우에서 데이터 스트림 클러스터링 분석 방법을 제안한다. 기존의 분석방법들은 모든 데이터 객체를 저장하고, 윈도우 범위가 지나면 각각 삭제하므로 두 배 이상으로 데이터를 처리해야 했다. 이를 개선하기 위해 윈도우 내 각 부분 통계정보를 가지는 페인 방법을 제시한다. 데이터 객체에 대한 각각의 처리 없이 페인 내의 분포정보를 기반으로 슬라이딩 윈도우를 갱신하여 데이터 처리의 효율성을 높인다. 실험을 통해 페인을 사용한 방법이 특정 구간에서 최대 50%까지 수행속도가 개선됨을 확인하였다.

Abstract

In a real-time data stream environment, clustering analysis does not efficiently reflect analysis results based on recent data due to the influence of past information. In clustering analysis, there are cases in which results that reflect insignificant past information are found. In such an environment, there is a sliding window technique as a method for classifying and processing past information. In this paper, we propose a data stream clustering analysis method in a sliding window. Conventional methods store all data objects and delete each data stream when the window range has passed, so it was necessary to process data twice or more. We propose sliding window panes with statistical information of each part within a window. By updating the sliding window based on the distribution information in the pane without processing each data object, the efficiency of data processing can be improved. Through the experiment, it was confirmed that our method improved the performance by up to 50% in a specific section compared to the existing method.

Keywords
clustering, data streams, sliding window panes, data mining

* 안양대학교 융합소프트웨어학과
- ORCID: http://orcid.org/0000-0002-3716-5760

· Received: Nov. 19, 2021, Revised: Dec. 26, 2021, Accepted: Dec. 29, 2021
· Corresponding Author: Namhun Park
Dept. of Convergence Software, Incheon Ganghwa-gun Bureun-myeon Jungang-ro 602-14, Korea,
Tel.: +82-32-930-6211, Email: nmhnpark@anyang.ac.kr

Ⅰ. 서 론

최근 스마트기기들이 활용되며 IT산업환경이 인터넷 스트림, 실시간 멀티미디어 데이터, 센서 데이터 스트림 등의 실시간으로 생성되는 방대한 데이터를 대상으로 변화함에 따라, 이러한 데이터 스트림을 대상으로 분석을 수행하는 데이터 마이닝 기법이 활발히 연구되고 있다.

데이터 스트림에 대한 클러스터링 분석 알고리즘은 다음 사항들을 고려해야 한다[1][2]. 데이터 스트림이 진행되면서 클러스터의 수가 계속 변화하므로, 기존 방법과 달리 수행 전에 분석할 클러스터의 수를 미리 정의할 수 없다. 또한, 중심점 기반으로 클러스터를 찾는 기존 방법에 비해 임의 모양의 클러스터를 식별할 수 있어야 한다. 예를 들어 네트워크 모니터링 환경에서, 연결의 분포는 매우 불규칙적이며 환경 관측 데이터에서도 유사한 환경 조건을 갖는 영역은 주로 임의의 모양으로 나타난다. 또한, 데이트 스트림 내의 이상데이터(Outlier)를 효과적으로 처리할 수 있어야 한다. 데이터 스트림의 데이터 항목들은 대개 순간적으로 생성되어, 일시적인 오류, 센서의 약한 배터리 전력 등과 같은 다양한 요인의 영향으로 일부 랜덤 잡음이 나타난다. 이들은 이상데이터로 클러스터 결과에 악영향을 줄 수 있다.

실시간 데이터 스트림을 대상으로 클러스터링을 수행하는 경우, 과거 정보에 영향을 받아 결과 클러스터는 데이터 스트림 내의 최신 정보를 효율적으로 반영하지 못한다. 해당 응용환경에서 현재는 중요하지 않은 과거 정보가 반영된 클러스터 결과를 찾는 경우가 발생한다. 이러한 데이터 스트림에서 과거의 정보를 구분하여 처리하기 위한 방법으로 슬라이딩 윈도우 기법이 있다. 슬라이딩 윈도우에서 오래된 객체들은 새로운 객체가 계속해서 도착하는 동안 소멸되어, 과거 객체들의 영향을 제거하고 새로운 객체들만을 처리하는 방법을 사용한다. 따라서, 소멸된 객체들의 영향을 제거하고 새로운 항목들을 슬라이딩 윈도우 구조에 병합시키는 효율적인 메커니즘을 제공하는 것은 아주 중요하다.

본 논문에서 슬라이딩 윈도우를 이용하여 데이터 스트림을 클러스터링 하는 방법을 제안한다. 셀트리[3] 방식의 클러스터링을 이용하면서 데이터 스트림을 슬라이딩 윈도우 처리하기 위해 패인 방법을 제안한다. 페인 방식에서는 슬라이딩 윈도우 범위를 정의된 수의 작은 크기의 페인으로 분할한다. 셀트리의 각 노드에 이러한 페인을 두어 해당 객체의 분포정보를 갱신한다. 새로운 데이터 항목이 입력되면 오래된 페인은 제거하고, 새로운 데이터의 페인을 생성하여, 각 페인이 슬라이딩 윈도우의 처리 단위가 된다. 슬라이딩 윈도우처럼 데이터 스트림 객체들을 저장하고, 소멸시키지만, 각 페인의 크기가 큰 경우에는, 슬라이딩 윈도우의 진행 단위가 증가함에 따라 페인 내의 객체 분포정보들도 점점 근사값이 된다. 정밀하게 파라미터를 설정하여 어느 정도 정확도를 유지하면서 페인을 수행할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서는 관련된 기존 연구들에 대해 설명하고, 3장에서는 셀트리 방식의 단계를 간략하게 살펴본다. 4장에서는 본 논문에서 제안한 데이터 스트림에 슬라이딩 윈도우를 적용시켜 클러스터를 생성하는 방식을 설명한다. 5장에서는 실험방법을 설명하고 6장에서 결론을 제시한다.

Ⅱ. 관련 연구

기존 데이터 스트림 클러스터링에 대한 연구로는 LSEARCH[4] 기법이 있다. 스트림을 작은 부분 데이터 스트림으로 나누어 각각을 대상으로 분할 기반 클러스터링을 수행하여 k개의 클러스터 중심을 구한다. 이후 모든 부분 데이터 스트림의 k개 클러스터 중심들에 대해 다시 클러스터링 과정을 반복하여 전체 데이터 스트림의 클러스터 중심들을 구한다. LSEARCH의 경우, 적절한 클러스터의 수 k를 쉽게 찾기가 어려우며, 이를 해결하기 위해 클러스터의 품질이 최대가 될 때까지 반복해서 수행해야 한다. VFKM[5] 알고리즘은 기존의 k-means 알고리즘을 확장한 것으로 데이터 스트림을 대상으로 적절한 수의 클러스터를 찾도록 개선하였다.

Hoeffding 경계를 적용하여 각 수행 단계에서 적절한 클러스터 수를 결정한다. 하지만, 데이터 스트림 응용환경에서 적절한 k가 스트림이 진행되면서

변화하는 경우 이를 처리할 수 없다.

CluStream[6]은 이전 연구에 비해 데이터 스트림에서 이러한 클러스터 변화를 탐색하기 위해 제안되었다. 이 기법은 먼저 k-means 알고리즘을 수행하여 극소 클러스터들을 생성한다. 클러스터의 그룹 구조를 가능한 정확히 발견하기 위해 극소 클러스터의 수는 메모리 공간에서 사용가능한 최대의 수로 정해진다. 오프라인 과정에서 CluStream은 극소 클러스터 결과들이 누적된 집합에 대해 k-means 알고리즘을 수행하여 최종 k개의 최종 매크로 클러스터를 찾는다. 그러나 이 알고리즘은 오프라인으로 매크로 클러스터를 찾는 방법으로 온라인 데이터 스트림의 실시간 클러스터를 발견하는 데는 효율적이지 않다.

이러한 k-means 기반의 분할기반 클러스터링 방식들은 임의모양의 클러스터를 찾는데 불리하다. 셀트리[3]는 격자기반 클러스터링 방식으로 조밀한 격자셀을 사용하여 임의모양의 클러스터 영역을 잘 찾아낼 수 있다. 본 논문에서는 셀트리를 개선하여 슬라이딩 윈도우 환경에서의 클러스터링 방법을 제시한다. 기존 연구들이 슬라이딩 윈도우 범위 경계의 각 데이터들을 반복적으로 처리해야하는데 비해, 본 논문에서는 격자셀 내에 페인을 구성하여 윈도우 범위별로 분포정보를 관리한다. 각 데이터에 대한 처리없이 페인 단위의 분포 정보를 이용하여 슬라이딩 윈도우 방법을 적용할 수 있어, 데이터 처리 성능에서 효율성을 지닌다. 또한 페인은 세부 공간 영역 내에서 구간 별 분포 통계를 관리하여 정밀한 크기로 설정할수록 오차없이 기존 알고리즘만큼의 정확도를 유지할 수 있다.

III. 데이터 스트림 클러스터링

다차원 데이터 공간 $N=N_1\times\cdots\times N_d$에서의 데이터 스트림은 다음과 같이 지속적으로 생성되는 데이터 항목의 무한집합으로 정의된다. t번째 순서에서 생성된 데이터 항목은 $e^t=<e_1^t,e_2^t,\cdots,ed^t>$, $ei^t\in N_i$, $1\le i\le d$로 표시한다. 현재의 데이터 스트림 D^t는 지금까지 생성된 모든 데이터 항목을 표시한다. 즉, $D^t=\{e_1, e_2,\cdots,e_t\}$이며, D^t에서 생성된 가중치 τ인 데이터 항목의 수는 $|D^t|$로 표시하면 식 (1)과 같다.

$$|D^t| = \begin{cases} 1 & \text{if } t \ge 2 \\ |D|^{t-1}\times\tau+1 & \text{if } t \ge 2 \end{cases} \quad (1)$$

특정 차원의 데이터 공간 N에서 현재의 데이터 스트림 D^t에 대해, $g(I, c^t, \mu^t, \sigma^t)$는 격자셀 g범위의 최근 데이터 분포에 대한 통계정보를 나타낸다. 이때 g는 데이터 공간 N의 영역에서 구간 I=[s,f)로 정의된다. D_g^t를 격자셀 g내의 D^t에 있는 항목들로 표시한다. 즉, $D_g^t=\{ e_i| e_i\in D^t \text{ and } e_i \in I \}$이다. 격자셀 g의 최근 분포에 대한 통계정보는 다음과 같이 정의된다.

c^t : D_g^t내의 감쇄율을 적용한 데이터 항목의 수.

$$c^t= \sum_{i=1}^{t}\tau^{(t-i)}$$

μ^t : μ^t 는 D_g^t내의 감쇄율을 적용한 데이터 항목들의 평균.

$$\mu^t= \sum_{i=1}^{t}e^i \times \tau^{(t-i)}/c^t$$

σi^t : σi^t 는 D_g^t내의 가중치를 적용한 데이터 항목들의 표준편차.

$$\sigma^t=\sqrt{\sum_{i=1}^{t}(e^i \times \tau^{(t-i)} - \mu^t)^2/c^t}$$

데이터 공간 N에서 현재 데이터 스트림 D^t가 주어졌을 때, 엔트리 수 m의 격자셀 리스트S는 격자셀 리스트 $S=<E_1,E_2,E_3,\cdots,E_p>$는 엔트리 $E_1,E_2,\cdots,E_p$의 단독 연결 리스트로 정의된다. 각 엔트리 Ei는 데이터 구조 E(min,max,G[1,⋯,m],next_ptr)를 유지하며, G[1,⋯,m]내의 m개의 엔트리는 최대 m개의 격자셀을 유지할 수 저장할 수 있다.

다차원 데이터 공간 $N=N_1\times\cdots\times N_d$의 데이터 스트림이 주어졌을 때, k차원($k \le d$) 격자셀의 영역은 직사각형 공간 $RS=I_1\times I_2\times\cdots\times I_k$로 정의된다. 여기에서 I_i는 차원 N_i, $1\le i\le k$의 구간이다. 이와 같은 직사각형 공간 내의 데이터 항목의 분포에 대한 통계정보를 효율적으로 모니터하기 위해 셀트리를 이용한다.

셀트리의 각 노드는 엔트리 E(min,max,G[1,⋯,m], next_ptr)과 자식노드배열 U[1,⋯,m]을 유지한다. 데이터 공간 $N=N_1\times\cdots\times N_d$의 현재 데이터 스트림 D^t

에 대해 엔트리 m인 셀트리는 엔트리 E<min,max, G[1,⋯,m],next_ptr>, 자식노드 포인터 U[1 ,⋯,m]로 구성된다. 여기에서 격자셀 G[i]의 자식 노드는 U[i], 1≤i≤m로 가리키며 그림 1과 같은 트리 구조를 가진다.

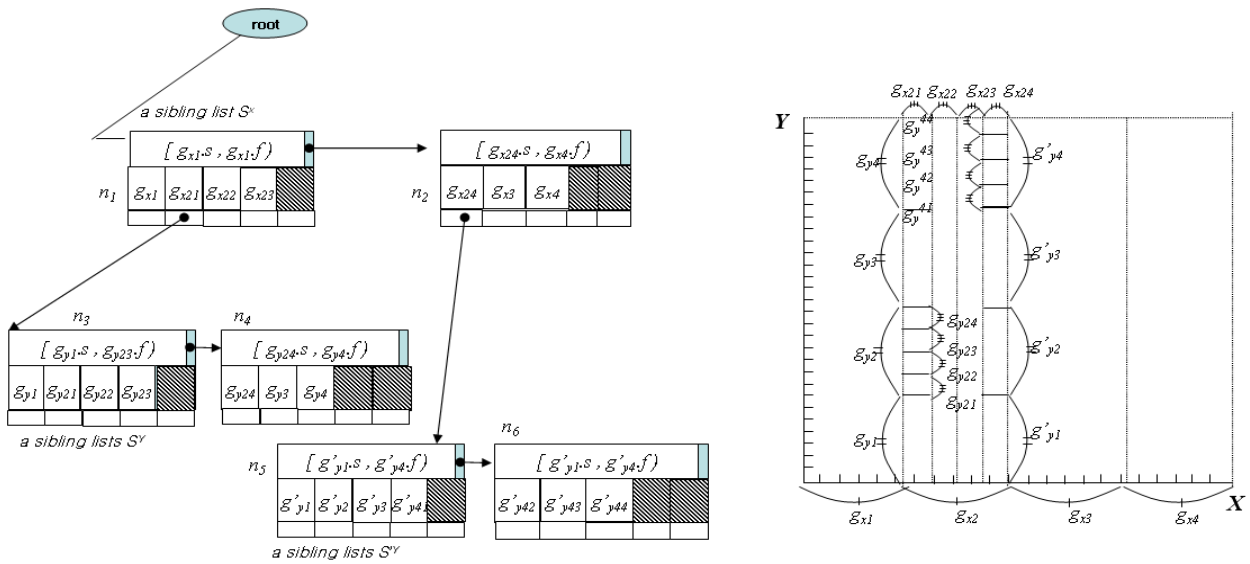
Ⅳ. 슬라이딩 윈도우

가중치를 적용하는 셀트리는 데이터 스트림을 클러스터링 하기 위해 제안되었다. 그러나 슬라이딩 윈도우와 달리 가중치 적용방식이 데이터 항목의 비중을 오래될수록 지수적으로 감소시키므로 클러스터들이 슬라이딩 윈도우의 범위에 남아있을 때도 셀트리에서는 빠르게 소멸될 수 있다.

슬라이딩 윈도에 대해서 클러스터링을 구현하면 슬라이딩 윈도우의 통계정보는 슬라이딩 윈도우를 들어오고 나갈 때 데이터 항목을 두 번 조사하여 모니터링 할 수 있다. 슬라이딩 윈도우 크기 내의 데이터 항목은 이후 소멸을 위해 유지된다. 새로운 데이터 항목이 도착하면 셀트리는 갱신되고, 갱신된 데이터 항목은 소멸될 때까지 순서대로 De-queue에 의해 유지된다. 수행되는 알고리즘을 살펴보면 그림 2와 같다.

```
1: function traverse(n, e^t, e^t-|w|, k)
   /*   n : 현재 탐색중인 노드      */
   /*   e^t : 최근 생성된 데이터     */
   /*   k  : 현재 탐색 레벨         */
   /*   e^t-|w| : dQ^t에서의 소멸항목 */

      /* 격자셀과 페인의 갱신 */
2:  find a grid-cell n.E.G[i] who includes e_k^t
3:  update the distribution statistics of n.E.G[i];
4:  update n.E.G[i].P^n;
5:  find a grid-cell n.E.G[j] who includes e_k^t-|w|
6:  remove previous panes n.E.G[j].P^n from e_k^t-|w|

      /* 셀트리 확장 및 생성 단계 */
7:  if (n.E.G[i].c^t/|D^t|≥ S_min and | n.E.G[i].I|==λ)
8:      do expanding as a cell-tree do;

      /* 셀트리 수축 단계 */
9:  else if (n.E.G[i]. c^t/|D^t|≤ S_mer)
10:    do shrinking as a cell-tree do;

      /* 다음 차원의 격자셀과 페인 탐색 */
11: if (n.E.U[i] ≠ null)
       traverse(n.E.U[i], e^t, k+1);

12: end // function end

13: main(root, D^t)
/*      root : a root of a cell tree   */
14:     for a newly generated data element e^t do
15:             update dQ^t with e_k^t;
16:             traverse(root, e^t, e^t-|w|, 1);
17:             t=t+1;
18:     end
19: end
```

그림 2. 페인 알고리즘
Fig. 2. Pane algorithm



(a) 셀트리 구조 (b) XxY 데이터공간

그림 1. 다차원 셀트리
Fig. 1. Multi-dimensional cell-tree, (a) Structure of the cell tree, (b) Data space XxY

데이터 스트림 D^t에 대해 슬라이딩 윈도우w가 주어지고, De-queue dQ^t 는 슬라이딩 윈도우 내의 데이터 항목들의 집합, 즉 $dQ^t=\{e^t-|w|,\cdots,e^t-1\}$으로 정의된다. 최근 데이터 스트림의 객체 e^t에 대해 먼저 dQ^t에 이를 반영하고(그림 2의 라인15), 격자셀의 분포와 페인에 e^t를 반영하고, dQ^t에서 소멸되는 $e^t-|w|$ 의 격자셀과 해당 페인을 탐색한다(라인16).

셀트리에서 e^t에 대한 격자셀의 분포정보를 갱신한 후(라인2-3), $e^t-|w|$의 분포정보를 해당 페인을 찾아 제거한다(라인5-6). 데이터 항목 $e^t-|w|$가 소멸될 때, $e^t-|w|$를 포함하는 격자셀 g는 새로운 데이터 항목 e^t에 의해 셀트리의 루트부터 조사된다. $e^t-|w|$ 의 경로에서 모든 격자셀 $e^t-|w|$의 분포정보는 다음과 같이 $e^t-|w|$ 를 빼어 g^t 가 된다.

$$g.c^t = g.c^v-1 \ , \ g.\mu_i^t=\frac{g.\mu_i^v \times g.c^v - e^{t-|w|}}{g.c^t} \qquad (2)$$

$$g.\sigma_i^t= \sqrt{\frac{g.c^v \times (g.\sigma^v)^2 - (e^{t-|w|})^2}{g.c^t}+\frac{(g.\mu_i^v)^2-(e^{t-|w|})^2}{g.c^t}-(g.\mu_i^t)^2}$$

셀트리 방식에서 각 격자셀은 그 영역에 있는 데이터 항목들의 분포정보를 유지한다. 본 논문에서는 슬라이딩 윈도우의 기간을 페인이라 부르는 h개의 서로 겹치지 않는 구간으로 나눈다. 각 페인은 그림 3과 같이 대응하는 구간 내의 데이터 항목들의 분포정보를 유지한다. 현재 데이터 스트림 D^t에 대해 슬라이딩 윈도우 w^t 가 주어지면 페인 p^n ($1\le n\le P$, P는 페인의 최대 수)은 그 구간 내에 데이터 항목의 분포정보 $p^n(c^t, \mu^t\ \sigma^t, [ts^k, tf^k))$를 다음과 같이 유지한다(라인 3-4)

$$c^t=|D_{wg}^t|, \quad \mu^t=\sum_{i=1}^{t} e^i/c^t, \quad \sigma^t=\sqrt{\sum_{i=1}^{t}(e^i-\mu^t)^2/c^t} \qquad (3)$$

슬라이딩 윈도우 w^t의 데이터 스트림 D_{wg}^t = $\{e^t|ts^n\le t\le tf^n,\ e^t \in g.I\}$에 대해 페인 p^n의 구간은 $[ts^n, tf^n)$, 크기는 $|w|/p$가 된다.

슬라이딩 윈도우 w^t에서 새로 생성된 페인의 모니터링 구간은 $[t, t+|w|/m\ )$ 가 된다. $[t, t+|w|/m\ )$구간에서 시각 t에서의 입력 데이터 항목들의 통계정보를 유지한다. 그것은 t에서 t+|w|/m로 들어오는 데이터 항목의 통계정보를 유지한다. 동시에 윈도우 밖으로 나가는 오래된 구간은 안 쓰이게 된다. 격자셀 $g(c^t, \mu^t, \sigma^t)$의 통계정보는 다음과 같이 페인의 통계정보 $p^i(c^t, \mu^t, \sigma^t)$ 로 부터 결정된다.

$$g.c^t=\sum_{i=1}^{P} p^i.c^t, \quad g.\mu^t=\sum_{i=1}^{P}(p^i.c^t \times p^i.\mu^t)/g.c^t \qquad (4)$$

$$g.\sigma^t= \sqrt{\frac{\sum_{i=1}^{P} p^i.c^t \times (p^i.\sigma^t)^2}{g.c^t}+\frac{\sum_{i=1}^{P}(p^i.\mu^t)^2}{g.c^t}-\sum_{i=1}^{P}(p^i.\mu_i^t)^2}$$

Ⅴ. 실 험

ENCLUS[7]에 의해 생성된 데이터 집합을 가지고 페인 메커니즘의 효율성을 증명하였다. ENCLUS 데이터는 지도에서 2차원 좌표값과 같이 다차원의 속성에 다수의 좌표값들로 구성된 데이터이다. 10차원의 1,000,000개의 객체로 이루어진 데이터 집합을 생성하였으며, 대부분의 데이터 항목은 랜덤하게 생성된 10개의 범위에 집중되어 클러스터를 구성하는데 범위의 폭 또한 랜덤하게 생성하였다. 모든 실험에서 데이터 항목은 온라인 데이터 스트림 환경을 실험하기 위해 순서대로 하나씩 처리되었다.

제안한 방식의 정확도를 그림 3에 나타내었다. CLIQUE[8]는 잘 알려진 유한 데이터 집합을 위한 전통적인 격자기반 클러스터링 알고리즘으로 제안한 방식의 정확성을 측정하기 위해 비교척도로 사용하였다. 제안한 방식으로 그룹화된 클러스터의 데이터 항목 중에서 CLIQUE에 의해 똑같은 클러스터로 그룹화된 데이터 항목들을 동일하게 클러스터링된 데이터 항목으로 정의하였다.

그림 4(a)는 새로운 데이터 항목이 생성될 때 제안한 방식의 정확성을 보여준다. 그림 4(b)는 윈도우 크기가 변할 때 정확도의 변화를 보여준다. 제안된 방식에서 페인의 수 h의 값이 변할 때, 정확도 수렴이 h에 따라 다양해진다. h에 의해 페인의 크기가 커질수록 더 근사치가 되며, 정확한 클러스터를 검색하는데 더 많은 시간을 필요로 하게 된다.

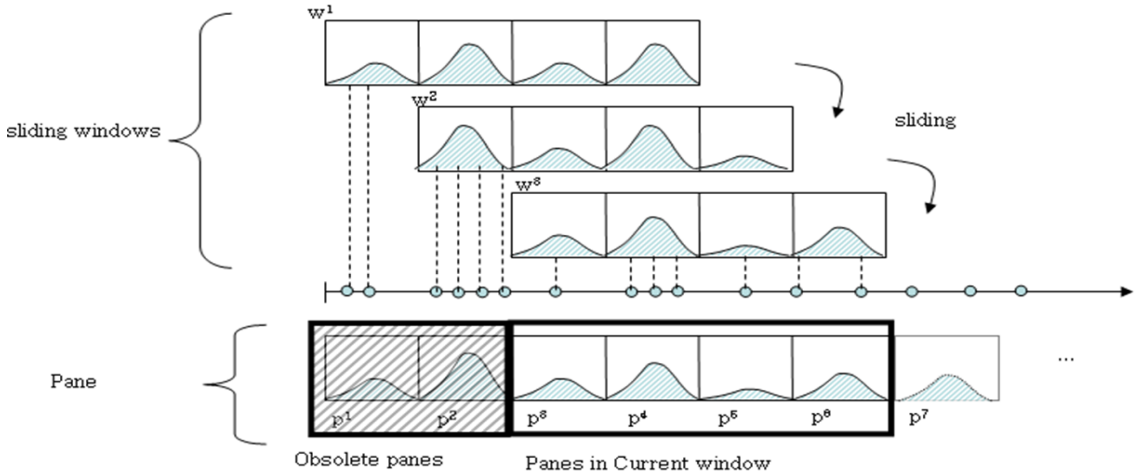
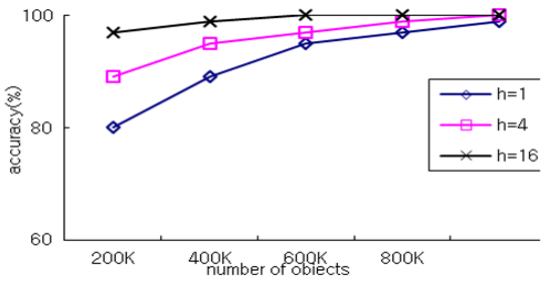
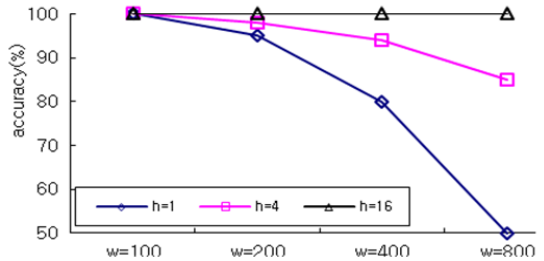


그림 3. 슬라이딩 윈도우에서의 페인 기법
Fig. 3. Pane method with sliding windows



(a) 페인 수에 따른 정확도
(a) Accuracy with the number of panes



(b) 윈도우 크기에 따른 정확도
(b) Accuracy with the size of windows
그림 4. 슬라이딩 윈도우 페인에서의 정확도
Fig. 4. Accuracy with sliding window panes

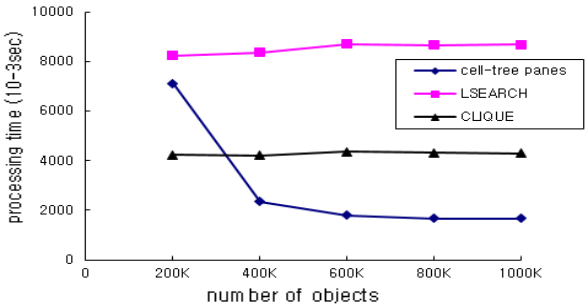


그림 5. 수행 시간 비교
Fig. 5. Comparison of processing times

그림 5에서는 기존의 LSEARCH, CLIQUE 방법과 수행시간을 비교하였다. LSEARCH 방법의 경우 데이터 스트림이 진행되면서 각 부분 스트림에 대해 k-means 기반의 탐색을 수행하며, 적절한 k를 찾는 과정을 수행하면서 다시 탐색을 반복하기 때문에 상대적으로 수행시간이 느리다. CLIQUE의 경우 다차원 격자구조를 모두 생성하여 데이터를 저장하는 방법으로 데이터 차원이 클수록 공간의 사용이 지수함수에 비례하여 증가한다.

이에 비해 수행속도는 빠르지만, 모든 객체를 각각 처리해야 한다. 페인 방법은 격자셀을 탐색하고 분할하는 과정에서는 잦은 분할과 분포정보 갱신으로 수행속도가 느리다. 하지만, 클러스터를 찾으면서부터 더 이상 불필요한 분할이 없어지며, 격자셀이 안정화되어 CLIQUE에 비해 적은 수의 격자셀로 빠르게 수행할 수 있다.

Ⅵ. 결 론

실세계 응용환경에서 데이터 스트림이 지속적으로 생성될 때, 유용한 정보를 얻거나 분석하기 위해서는 과거보다 최근 생성된 데이터의 처리가 훨씬 중요하다. 본 논문에서는 최근 데이터를 효율적으로 분석하기 위해 슬라이딩 윈도우 환경에서의 클러스터링 방법을 제안하였다. 페인 방법을 사용하여 슬라이딩 윈도우의 기간을 미리 정해진 수만큼 동일한 크기의 페인으로 나눈다. 각 노드에서는 해당 영

역에 대한 페인을 유지하여 데이터의 분포정보를 관리한다. 윈도우가 슬라이딩될 때, 각 페인은 슬라이딩 윈도우를 위한 단위가 되어, 데이터 각각을 처리하는 기존 분석방법보다 효율적으로 데이터 스트림을 처리할 수 있다.

실험에서 기존 방법과 비교하였을 때, 초기 분할 과정에서 수행시간이 다른 방법들보다 오래 걸렸지만 점점 분할횟수가 줄어들면서 수행속도가 안정화되어 다른 방법들보다 최대 50%까지 향상됨을 볼 수 있었다. 또한, 정확도 실험을 통해 데이터 각각을 처리하는 방법만큼 정확도가 유지되도록 파라미터를 조절할 수 있어, 실시간 데이터 스트림 환경에서도 기존 방법만큼의 정확도를 유지하며 빠른 성능으로 수행할 수 있었다.

References

[1] Ming Hua, Jian Pei, and Xuemin Lin, "Ranking queries on uncertain data", The International Journal on Very Large Data Bases, Vol. 20, No. 1, pp. 129-153, Feb. 2011. http://dx.doi.org/10.1007/s00778-010-0196-4.

[2] Jie Zhao, Xiaowen Li, and Peiquan Jin, "A Time-Enhanced Topic Clustering Approach for News Web Search", Int. Journal of Database Theory and Application, Vol. 5, No. 4, pp. 1-10, Dec. 2012. https://www.earticle.net/Article/A207866.

[3] Nam Hun Park and Won Suk Lee, "Cell trees: An Adaptive Synopsis structure for clustering multi-dimensional on-line data streams", J. Data & Knowledge Engineering, Vol. 63, No. 2, pp. 528-549, Nov. 2007. http://dx.doi.org/10.1016/j.datak.2007.04.003.

[4] Jonathan de Andrade Silva and Eduardo Raul Hruschka, "A Support System for Clustering Data Streams with a Variable Number of Clusters", ACM Transactions on Autonomous and Adaptive Systems, Vol. 11, No. 2, pp 1-26, Jul. 2016. https://doi.org/10.1145/2932704.

[5] Mohamed Medhat Gaber, "Advances in data stream mining", Data Mining and Knowledge Discovery, Vol. 2, No. 1, pp. 79-85, Nov. 2011. https://doi.org/10.1002/widm.52.

[6] Eoin Martino Grua, Mark Hoogendoorn, Ivano Malavolta, Patricia Lago, and A.E. Eiben, "Clu StreamGT Online Clustering for Personalization in the Health Domain", 2019 IEEE/WIC/ACM International Conference on Web Intelligence, Thessaloniki, Greece, pp. 270-275, Oct. 2019. https://doi.org/10.1145/3350546.3352529.

[7] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang, "Entropy-based subspace clustering for mining numerical data", In Proceedings of the fifth ACM SIGKDD International Conference on Knowledge discovery and data mining, San Diego California, USA, pp. 84-93, Aug. 1999. https://doi.org/10.1145/312129.312199.

[8] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications", In Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, pp. 94-105, Jun. 1998. https://doi.org/10.1145/276304.276314.

저자소개

박 남 훈 (Namhun Park)

2000년 2월 : 연세대학교 기계전자공학부(공학사)
2002년 2월 : 연세대학교 컴퓨터과학과(공학석사)
2007년 8월 : 연세대학교 컴퓨터과학과(공학박사)
2010년 2월 ~ 현재 : 안양대학교 융합소프트웨어학과 교수
관심분야 : 데이터 마이닝, 데이터 스트림, 빅데이터